NL5
User's Reference

Components
Operators
Functions
Script commands

Ver.3.12

VERSION

NL5 User's Reference version 3.12.23, 04/24/2024
The latest versions of NL5 documents can be found at sidelinesoft.com/nl5.

LIMITED LIABILITY

COPYRIGHTS

sidelinesoft

Table of Contents

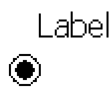
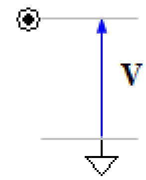
Components

Logical levels and threshold for all components are defined in *Schematic settings/Components* window.

SubCir model

Model	Parameter	Units	Description
SubCir	**File**		File name of subcircuit schematic.
	Pin1		Name of subcircuit label connected to pin 1
	…		**…**
	PinN		Name of subcircuit label connected to pin N
	Cmd		Subcircuit start-up command string
	IC		Subcircuit Initial conditions string
See *Working with Subcircuits* chapter of The NL5 Manual for details.			

Label

Symbol	Models			Signals
	Label V Step Single Pulse	Clock Sin Sweep Function List	File Trace IC SubCir	

All models (except SubCir) can be used:

- As a voltage trace probe point.
- For connecting schematic points without wires, including points at different sheets.

Model	Parameter	Units	Description
Label	No parameters.		

Label can be used:

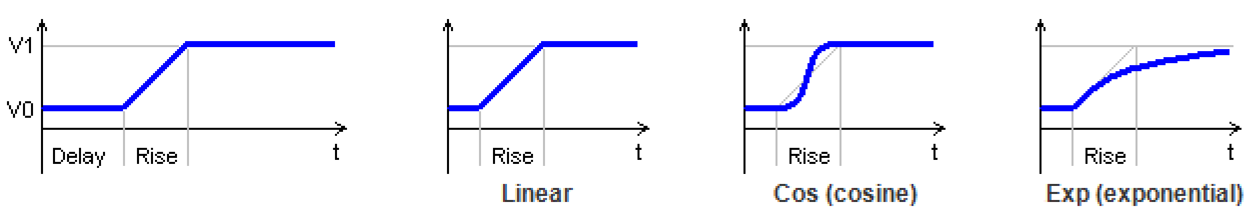
- As a voltage trace probe point.
- For connecting schematic points without wires, including points at different sheets.

Model	Parameter	Units	Description
V	**V**	V	Voltage.

Constant voltage = **V**.

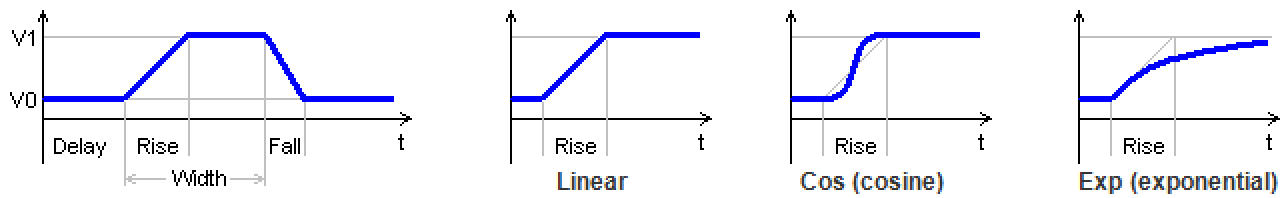
Model	Parameter	Units	Description
Step	**V1**	V	Step On voltage.
	V0	V	Step Off voltage.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

Step starts after **Delay** time. If **Rise** is non-zero, 3 **Slope** types are available.



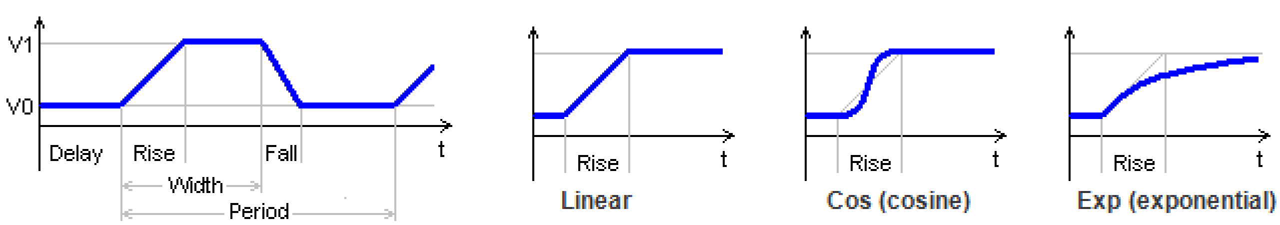
Model	Parameter	Units	Description
Single	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

Single pulse starts after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



Model	Parameter	Units	Description
Pulse	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

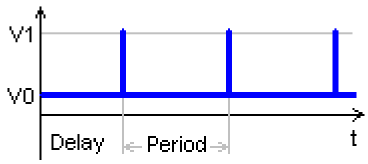
Pulses start after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



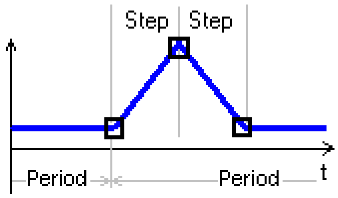
Model	Parameter	Units	Description
Clock	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

Periodic pulses with width of one simulation step. Pulses start after **Delay** time.

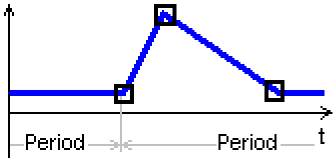
Clock model is recommended to produce a constant frequency clock signal for C-code, DLL, logical components, etc. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:

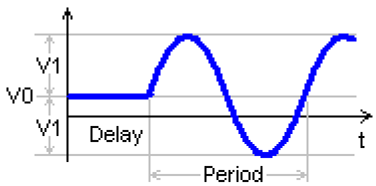


Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:

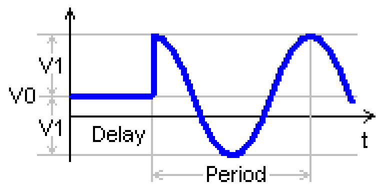


Model	Parameter	Units	Description
Sin	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

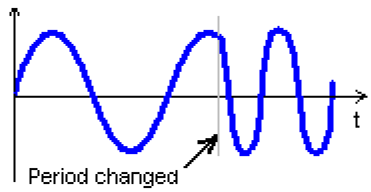
Sinusoidal signal starts after **Delay** time. **Phase** is sine phase in degrees at the moment when signal starts. If transient is paused, sine period changed, and then transient is continued, the phase of the signal remains continuous, providing smooth sine signal of variable frequency. If **Decay** is not zero, the sine signal is exponentially dumped with time constant = 1/**Decay**.



Phase = 0



Phase = 90



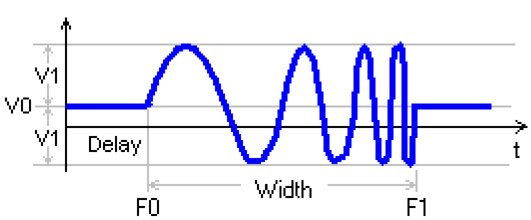
Model	Parameter	Units	Description
Sweep	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

Sinusoidal signal with variable frequency starts after **Delay** time. Signal frequency changes during **Width** interval from **F0** to **F1** linearly or exponentially, depending on specified **Type.**

If **F0** = **F1**, then one period of frequency 1/**Width** will be generated.

If lowest frequency is set to zero and **Type** = Exp, then lowest frequency 0.01/**Width** will be used.

If needed, the highest frequency will be increased to provide integer number of signal periods, so that signal phase at the beginning and at the end of **Width** interval is exactly zero.



Model	Parameter	Units	Description
Function	**f**	V	Function

Arbitrary function **f** defines voltage as a function of the following variables:

t **-** current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, voltage is zero.

Example:

f = sin(t) * (1+cos(t*.01))
f = V(R1) * I(R1)

Please note that **V**, **I**, **P**, and **S** variables are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined by **List** parameter in the **csv** (comma-separated values) format, as follows:

t0,V0,t1,V1,...,tn,Vn

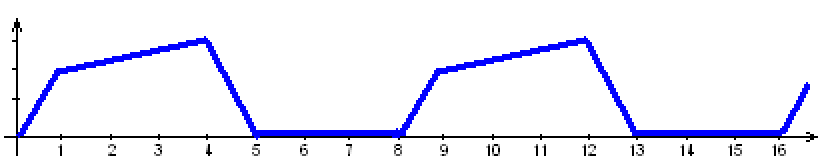
where all t and V can be numerical values or expressions.
If t<t0, signal is V0.
If t0<t<t1, signal value is linearly interpolated between V0 and V1, etc.
If t>tn, and **Cycle** parameter is set to **No**, the signal value is Vn. Otherwise the signal defined in t0…tn interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
List = 0,0,1,2,4,3,5,0,8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following voltage will be generated:



See *Working with List source* chapter for more details.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Signal is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored >
t0,V0
t1,V1
.....
tn,Vn
```

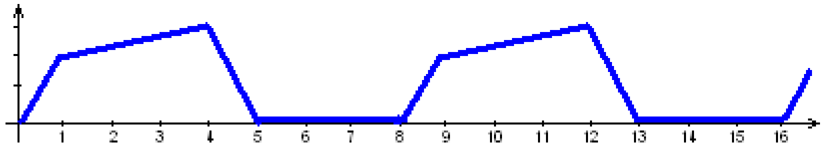
where all t and V can be numerical values or expressions.
If t<t0, signal is V0. If t0<t<t1, signal value is linearly interpolated between V0 and V1, etc. If t>tn, and **Cycle** parameter is set to **No**, the signal value is Vn. otherwise the signal defined in t0…tn interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example. File content:

```
0,0
1,2
4,3
5,0
8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following voltage will be generated:



Model	Parameter	Units	Description
Trace	**Trace**		Trace name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Signal is defined by an existing trace. **Trace** parameter is a name of a transient trace. Only traces loaded from data file, imported from text or binary file, duplicated, or pasted from the clipboard can be used.

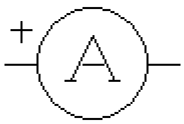
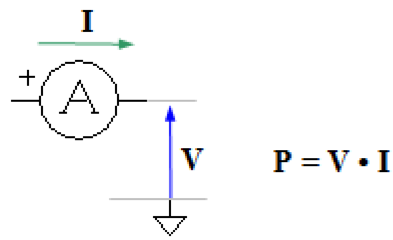
Signal start is delayed by **Delay** time.

If **Cycle** parameter is set to **Yes**, the signal is repeated continuously.

Model	Parameter	Units	Description
IC	**V**	V	Initial voltage.
	R	Ohm	Initial resistance.

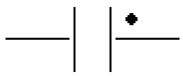
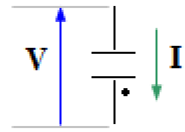
Initial condition. The model is used to apply initial voltage during DC operating point calculation. When calculating DC operating point, the temporary voltage source **V** is connected to the label through initial resistor **R**. When DC operating point is found, the voltage source is removed, and the **IC** model operates similar to **Label** model.

A – Amperemeter

Symbol	Models	Signals
	Amperemeter	

Model	Parameter	Units	Description
Amperemeter	No parameters.		
Short circuit. In addition to current, amperemeter can measure voltage relative to ground, and power delivered to grounded load.			

C – Capacitor

Symbol	Models	Signals
	C PWL SubCir	 $P = V \cdot I$

Model	Parameter	Units	Description
C	**C**	F	Capacitance
	IC	V	Initial condition: voltage. Leave blank if IC not defined.

Linear capacitor: I = **C** * dV/dt.

When calculating DC operating point, if **IC** is not blank, capacitor is replaced with voltage source equal to **IC**. Otherwise, capacitor is temporarily removed (open circuit), DC operating point is calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage.

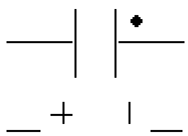
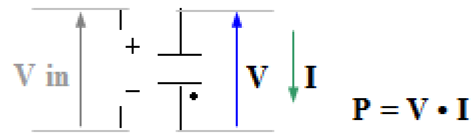
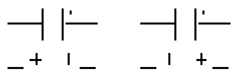
Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, C(V)
	IC	V	Initial condition: voltage. Leave blank if IC not defined.

Piecewise constant capacitor: **pwl** string defines capacitance as a function of voltage across the capacitor C(V). Capacitor charge Q(V) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that C(V) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

When calculating DC operating point, if **IC** is not blank, capacitor is replaced with voltage source equal to **IC**. Otherwise, capacitor is temporarily removed (open circuit), DC operating point is calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage.

C – Voltage controlled capacitor

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, C(V*in*)
	IC	V	Initial condition: voltage. Leave blank if IC not defined.

Piecewise constant voltage controlled capacitor. **pwl** string defines capacitance as a function of control voltage C(V*in*). At any moment:

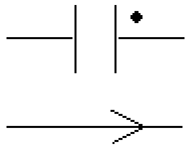
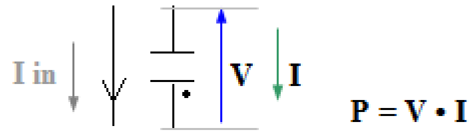
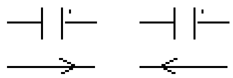
I = C(V*in*) * dV/dt.

See *Working with PWL model* chapter for details.

Please note that C(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

When calculating DC operating point, if **IC** is not blank, capacitor is replaced with voltage source equal to **IC**. Otherwise, the capacitor is temporarily removed (open circuit), DC operating point calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage.

C – Current controlled capacitor

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, C(I*in*)
	IC	V	Initial condition: voltage. Leave blank if IC not defined.

Piecewise constant current controlled capacitor. **pwl** string defines capacitance as a function of control current C(I*in*). At any moment:

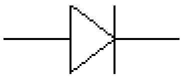
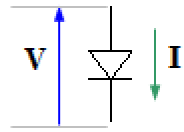
I = C(I*in*) * dV/dt.

See *Working with PWL model* chapter for details.

Please note that C(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

When calculating DC operating point, if **IC** is not blank, capacitor is replaced with voltage source equal to **IC**. Otherwise, the capacitor is temporarily removed (open circuit), DC operating point calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage

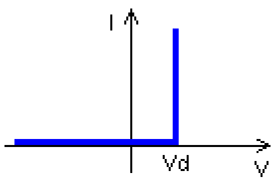
D – Diode

Symbol	Models	Signals
	Diode Storage Soft PWL SubCir	 $P = V \cdot I$

Model	Parameter	Units	Description
Diode	**Vd**	V	Forward voltage drop.
	IC		Initial condition: On/Off.

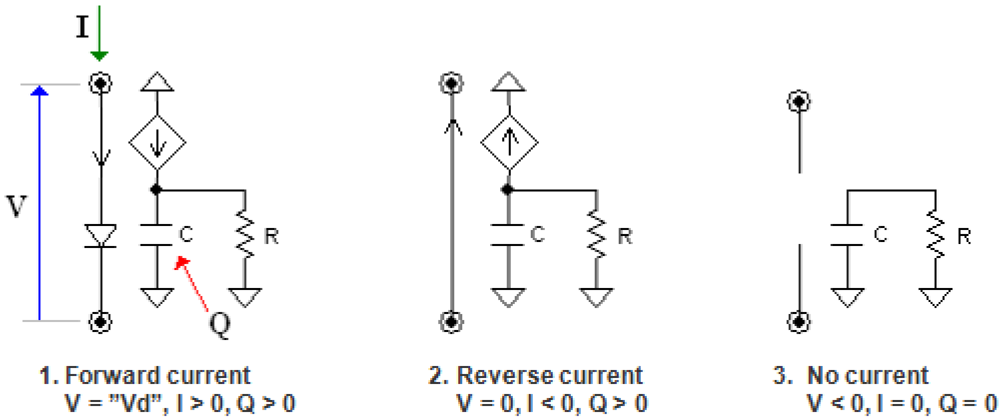
Ideal diode. If V >= **Vd**, diode is On (short circuit). Otherwise diode is Off (open circuit, I=0).

When calculating DC operating point diode is set to the state specified in **IC**.



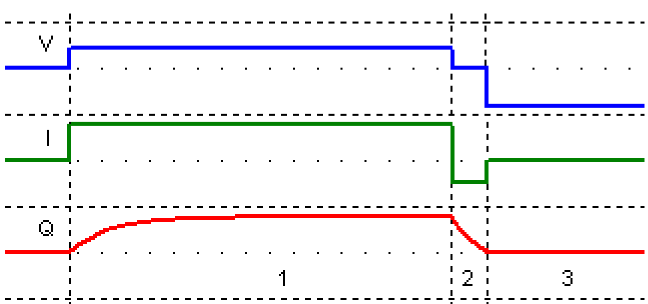
Model	Parameter	Units	Description
Storage	**Vd**	V	Forward voltage drop.
	t	s	Recombination time constant.
	IC		Initial condition: Off/On.
	ICQ	C (A*s)	Initial condition: charge.

Charge storage diode. Simplified equivalent schematic of the model is the following:



The diode has internal capacitor C and resistor R, with the time constant RC = **t**, Q is the charge on the capacitor.

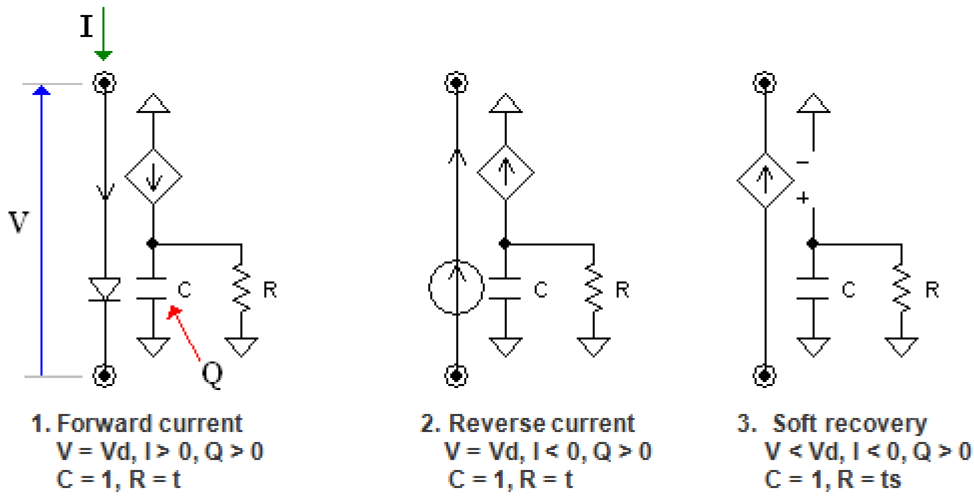
In **mode 1**, forward current flows through the diode and forward voltage drop is **Vd**. At the same time, the current equal to forward current is charging capacitor C. In **mode 2**, reverse current is applied to the diode, and capacitor C is being discharged by the current equal to reverse current. As long as charge Q on the capacitor is positive, the diode is a short circuit with zero voltage drop. Finally, when charge drops to zero, the diode switches to **mode 3**, with zero current and negative voltage drop (open circuit).



When calculating DC operating point the diode is set to the state specified in **IC**, and internal charge Q is set to specified **ICQ** value.

Model	Parameter	Units	Description
Soft	**Vd**	V	Forward voltage drop.
	t	s	Recombination time constant.
	ts	s	Soft recovery time constant.
	IC		Initial condition: Off/On.
	ICQ	C (A*s)	Initial condition: charge.

Soft recovery charge storage diode. Simplified equivalent schematic of the model is the following:



The diode has internal capacitor C=1 and resistor R. Time constant RC is equal either recombination time constant **t**, or soft recovery time constant= **ts**. Q is the charge on the capacitor. In **mode 1**, forward current flows through the diode and forward voltage drop is **Vd**. At the same time, the current equal to forward current is charging capacitor C. In **mode 2**, reverse current is applied to the diode, and capacitor C is being discharged by the current equal to reverse current. Voltage drop on the diode is still **Vd**. At the moment when reverse current is equal or less than charge divided by soft recovery time constant **ts**, a **mode 3** is turned on. In **mode 3**, capacitor C is being exponentially discharged by the current through resistor R with time constant **ts** (plus small constant current to ensure full discharge - not shown on the picture). Reverse diode current is proportional to the charge. As soon as charge drops to zero, the diode switches to **mode 4** (not shown), with zero current and negative voltage drop (open circuit).

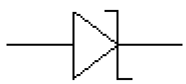
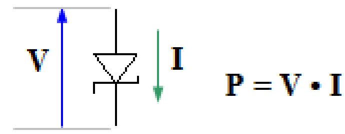
When calculating DC operating point the diode is set to the state specified in **IC**, and internal charge Q is set to specified **ICQ** value.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, R(V)

Piecewise linear diode. **pwl** string defines resistance as a function of voltage across the diode R(V). Volt-ampere characteristic of the diode is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that R(V) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

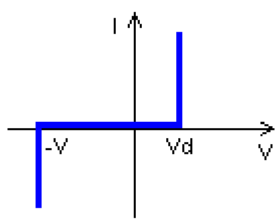
D – Zener

Symbol	Models	Signals
	Zener PWL SubCir	

Model	Parameter	Units	Description
Zener	**V**	V	Breakdown voltage drop.
	Vd	V	Forward voltage drop.
	IC		Initial condition: Minus/Off/Plus.

Ideal zener. If V (across zener) <= **-V** or V >= **Vd**, zener is On (short circuit). Otherwise zener is Off (open circuit, I=0).

When calculating DC operating point zener is set to the state specified in **IC**.

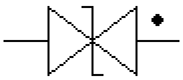
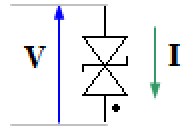


Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, R(V)

Piecewise linear zener. **pwl** string defines resistance as a function of voltage across zener R(V). Volt-ampere characteristic of zener is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that R(V) is **piecewise constant** function, although he model and parameter are still called **pwl** for historical reasons.

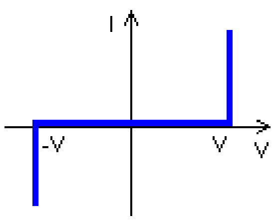
D – Bidirectional zener

Symbol	Models	Signals
	Zener PWL SubCir	 $P = V \cdot I$

Model	Parameter	Units	Description
Zener	**V**	V	Breakdown voltage drop.
	IC		Initial condition: Minus/Off/Plus.

Ideal bidirectional zener. If V (across zener) <= **-V** or V >= **V**, zener is On (short circuit). Otherwise zener is Off (open circuit, I=0).

When calculating DC operating point zener is set to the state specified in **IC**.

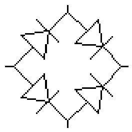


Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, R(V)

Piecewise linear zener. **pwl** string defines resistance as a function of voltage across zener R(V). Volt-ampere characteristic of zener is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that R(V) is **piecewise constant** function, although he model and parameter are still called **pwl** for historical reasons.

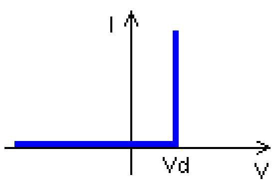
D – Bridge rectifier

Symbol	Models	Signals
	Diode	$P = V_1 \cdot I_1 + V_2 \cdot I_2 + V_3 \cdot I_3 + V_4 \cdot I_4$

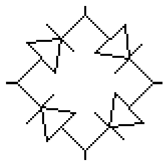
Model	Parameter	Units	Description
Diode	**Vd**	V	Forward voltage drop.

Bridge rectifier with ideal diodes. For each diode, if V >= **Vd**, diode is On (short circuit). Otherwise diode is Off (open circuit, I=0).

When calculating DC operating point all diodes are Off.



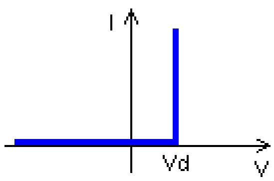
D – Diode ring

Symbol	Models	Signals
	Diode	$P = V_1 \cdot I_1 + V_2 \cdot I_2 + V_3 \cdot I_3 + V_4 \cdot I_4$

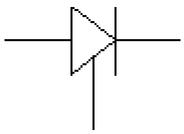
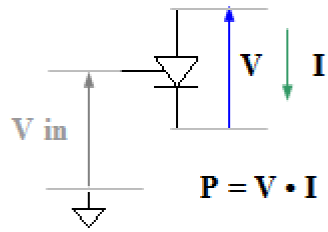
Model	Parameter	Units	Description
Diode	**Vd**	V	Forward voltage drop.

Diode ring with ideal diodes. For each diode, if V >= **Vd**, diode is On (short circuit). Otherwise diode is Off (open circuit, I=0).

When calculating DC operating point all diodes are Off.



D – Logic controlled thyristor

Symbol	Models	Signals
	Thyristor SubCir	

Model	Parameter	Units	Description
Thyristor	**Vd**	V	Forward voltage drop.
	Ihold	A	Holding current.
	IC		Initial condition: Off/On.

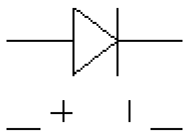
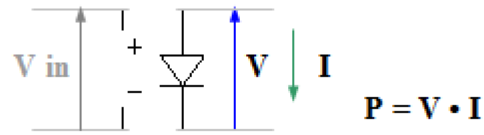
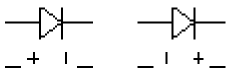
Thyristor has two states:

- Off state (non-conducting): open circuit.
- On state (conducting): ideal diode with **Vd** forward voltage drop.

If control voltage V*in* is greater than logical threshold, thyristor is in On state (ideal diode). When control voltage drops below logical threshold, thyristor stays in On state as long as current I exceeds holding current **Ihold**, and voltage V is not negative.

When calculating DC operating point thyristor is set to the state specified in **IC**.

D – Voltage controlled thyristor

Symbol	Models	Signals
	Thyristor SubCir	
Views		

Model	Parameter	Units	Description
Thyristor	**Vd**	V	Forward voltage drop.
	Ihold	A	Holding current.
	Threshold	V	Voltage threshold.
	IC		Initial condition: Off/On.

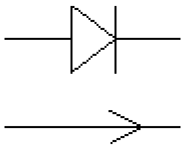
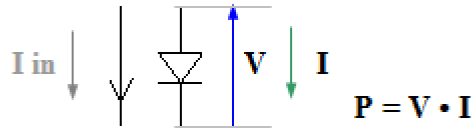
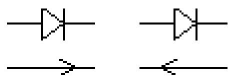
Thyristor has two states:

- Off state (non-conducting): open circuit.
- On state (conducting): ideal diode with **Vd** forward voltage drop.

If control voltage V*in* is greater than **Threshold**, thyristor is in On state (ideal diode). When control voltage drops below **Threshold**, thyristor stays in On state as long as current I exceeds holding current **Ihold**, and voltage V is not negative.

When calculating DC operating point thyristor is set to the state specified in **IC**.

D – Current controlled thyristor

Symbol	Models	Signals
	Thyristor SubCir	
Views		

Model	Parameter	Units	Description
Thyristor	**Vd**	V	Forward voltage drop.
	Ihold	A	Holding current.
	Threshold	A	Current threshold.
	IC		Initial condition: Off/On.

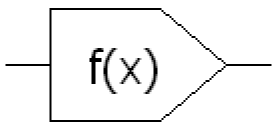
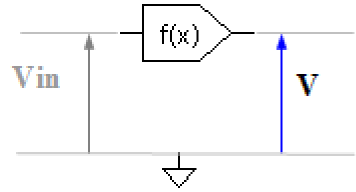
Thyristor has two states:

- Off state (non-conducting): open circuit.
- On state (conducting): ideal diode with **Vd** forward voltage drop.

If control current I*in* is greater than **Threshold**, thyristor is in On state (ideal diode). When control current drops below **Threshold**, thyristor stays in On state as long as current I exceeds holding current **Ihold**, and voltage V is not negative.

When calculating DC operating point thyristor is set to the state specified in **IC**.

F – Function

Symbol	Models	Signals
	Function Pwr Abs Int Lim Table SubCir	

The function is calculated and applied to the output on every calculation step.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Examples:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

Model	Parameter	Units	Description
Pwr	**power**		Power.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

"Signed" power function: V = **K** * pwr(V*in*, **power**).

The function is calculated as follows:

if **power** = 0:
- if V*in* < 0 . . . : V = **-K**
- if V*in* = 0 . . . : V = 0
- if V*in* > 0 . . . : V = **K**

if **power** ≠ 0:
- if V*in* < 0 . . . : V = **-K** * $(-V_{in})^{\mathbf{power}}$
- if V*in* = 0 . . . : V = 0
- if V*in* > 0 . . . : V = **K** * $V_{in}^{\mathbf{power}}$

Model	Parameter	Units	Description
Abs	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Absolute value: V = **K** * abs(V*in*).

Model	Parameter	Units	Description
Int	**resolution**	V	Resolution.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Rounding function: V = **K** * round(V*in*, **resolution**).

Round to the nearest multiple of **resolution**. If **resolution** = 1, round to the nearest integer.

Model	Parameter	Units	Description
Lim	**Max**	V	Maximum.
	Min	V	Minimum.
	IC	V	Initial condition: output voltage.

Limiting function is calculated as follows:

if V*in* < **Min** . . . : V = **Min**
if V*in* > **Max** . . : V = **Max**
Otherwise . . . . : V = V*in*

Model	Parameter	Units	Description
Table	**Table**		Comma-separated string, V*in*/V*out* pairs.
	IC	V	Initial condition: output voltage.

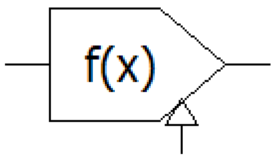
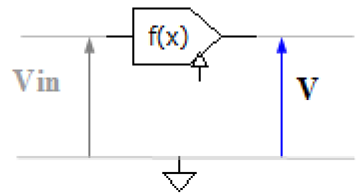
Look-up table. Function output is defined by **Table** parameter in the **csv** (comma separated values) format, as follows:

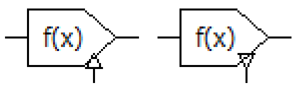
X1,Y1,X2,Y2,…,XN,YN

where Xi,Yi pair defines input value (X) and output value (Y). Output value between specified points is linearly interpolated. Output value below X1 is linearly extrapolated using X1…X2 interval data, output value above XN is linearly extrapolated using X(N-1)…XN interval data. Values X1…XN should be given in an ascending order.

See *Working with Table model* chapter for more details.

F – Function with clock

Symbol	Models	Signals
	Function Pwr Abs Int Lim Table SubCir	



Function component with **clock** operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Examples:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

Model	Parameter	Units	Description
Pwr	**power**		Power.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

"Signed" power function: V = **K** * pwr(V*in*, **power**).

The function is calculated as follows:

if **power** = 0:
- if V*in* < 0 . . . : V = **-K**
- if V*in* = 0 . . . : V = 0
- if V*in* > 0 . . . : V = **K**

if **power** ≠ 0:

- if V*in* < 0 . . . : V = **-K** * $(-V_{in})^{power}$
- if V*in* = 0 . . . : V = 0
- if V*in* > 0 . . . : V = **K** * V_{in}^{power}

Model	Parameter	Units	Description
Abs	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Absolute value: V = **K** * abs(V*in*).

Model	Parameter	Units	Description
Int	**resolution**	V	Resolution.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Rounding function: V = **K** * round(V*in*, **resolution**).

Round to the nearest multiple of **resolution**. If **resolution** = 1, round to the nearest integer.

Model	Parameter	Units	Description
Lim	**Max**	V	Maximum.
	Min	V	Minimum.
	IC	V	Initial condition: output voltage.

Limiting function is calculated as follows:

if V*in* < **Min** . . . : V = **Min**
if V*in* > **Max** . . : V = **Max**
Otherwise . . . . : V = V*in*

Model	Parameter	Units	Description
Table	**Table**		Comma-separated string, V*in*/V*out* pairs.
	IC	V	Initial condition: output voltage.

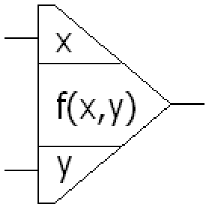
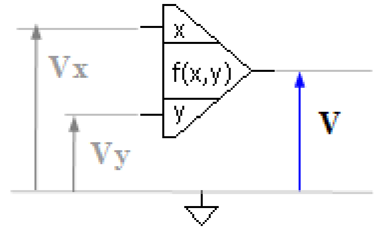
Look-up table. Function output is defined by **Table** parameter in the **csv** (comma separated values) format, as follows:

X1,Y1,X2,Y2,…,XN,YN

where Xi,Yi pair defines input value (X) and output value (Y). Output value between specified points is linearly interpolated. Output value below X1 is linearly extrapolated using X1…X2 interval data, output value above XN is linearly extrapolated using X(N-1)…XN interval data. Values X1…XN should be given in an ascending order.

See *Working with Table model* chapter for more details.

F – Function-2

Symbol	Models		Signals
	Function Mul Div Sum Sub Max Min	Pwr Mag Phase GT LT Table SubCir	

The function is calculated and applied to the output on every calculation step.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the inputs.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*x*
y – input voltage V*y*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = sqrt(x*x + y*y)
f = x * y * sin(t)
f = P(r1) + P(r2)

Model	Parameter	Units	Description
Mul	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.
Multiplication**:** V = **K** * V*x* * V*y*.			

Model	Parameter	Units	Description
Div	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.
Division**.:** V = **K** * V*x* / V*y*. If V*y* = 0, V = 0.			

Model	Parameter	Units	Description
Sum	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.
Addition: V = **K** * (V*x* + V*y*).			

Model	Parameter	Units	Description
Sub	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.
Subtraction: V = **K** * (V*x* - V*y*).			

Model	Parameter	Units	Description
Max	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.
Maximum: V = **K** * max(V*x*, V*y*). if V*x* >= V*y* . . : V = **K** * V*x* if V*x* < V*y* . . . : V = **K** * V*y*			

Model	Parameter	Units	Description
Min	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.
Maximum**:** V = **K** * max(V*x*, V*y*). if V*x* >= V*y* . . : V = **K** * V*x* if V*x* < V*y* . . . : V = **K** * V*y*			

Model	Parameter	Units	Description
Pwr	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

"Signed" power function**:** V = **K** * pwr(V*x*, V*y*).

The function is calculated as follows:

if V*y* = 0:
- if V*x* < 0. . . : V = **-K**
- if V*x* = 0. . . : V = 0
- if V*x* > 0. . . : V = **K**

if V*y* ≠ 0:
- if V*x* < 0. . . : V = **-** **K** * $(-Vx)^{Vy}$
- if V*x* = 0. . . : V = 0
- if V*x* > 0. . . : V = **K** * Vx^{Vy}

Model	Parameter	Units	Description
Mag	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Magnitude**.** V = **K** * sqrt(Vx^2 + Vy^2).

Model	Parameter	Units	Description
Phase	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Phase: V = **K** * phase(V*x*, V*y*).

V in Volts is equal to phase of a vector V*x* + jV*y* in degrees.
If V*x* = 0 and V*y* = 0: V = 0.

Model	Parameter	Units	Description
GT	**IC**	V	Initial condition: output voltage.

Greater than: V = V*x* > V*y* ? **High** : **Low**.

High and **Low** are logical levels.

Model	Parameter	Units	Description
LT	**IC**	V	Initial condition: output voltage.

Less than. V = V*x* < V*y* ? **High** : **Low**.

High and **Low** are logical levels.

Model	Parameter	Units	Description
Table	**X**		Comma-separated string, X (input values).
	Y		Comma-separated string, Y (input values).
	Table		Comma-separated string, Table of Z (output values).
	IC	V	Initial condition: output voltage.

2D look-up table. **Table** parameter defines output Z as a function of X and Y inputs of the component in the following format:

Z11,Z12,…,Z1N,Z21,Z22,…,Z2N,…,ZM1,ZM2,…,ZMN

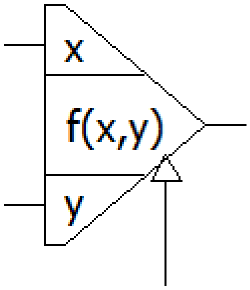
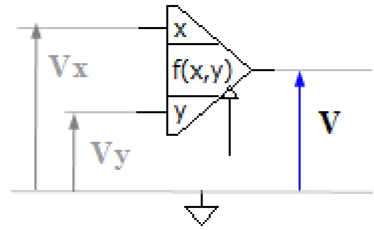
where:

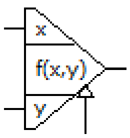
- Zij defines output of the function for input values Xi and Yj;
- N is total number of X input values, defined by **X** parameter;
- M is total number of Y input values, defined by **Y** parameter.

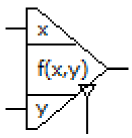
Output value between specified X and Y points is linearly interpolated on both coordinates. Output value below X1 is linearly extrapolated using X1…X2 interval data, output value above XN is linearly extrapolated using X(N-1)…XN interval data. The same rule is applied to Y coordinate

See *Working with 2D Table model* chapter for more details.

F – Function-2 with clock

Symbol	Models		Signals
	Function Mul Div Sum Sub Max Min	Pwr Mag Phase GT LT Table SubCir	





Function component with **clock** operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the inputs.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*x*
y – input voltage V*y*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = sqrt(x*x + y*y)
f = x * y * sin(t)
f = P(r1) + P(r2)

Model	Parameter	Units	Description
Mul	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Multiplication**:** V = **K** * Vx * Vy.

Model	Parameter	Units	Description
Div	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Division**.:** V = **K** * Vx / Vy. If Vy = 0, V = 0.

Model	Parameter	Units	Description
Sum	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Addition: V = **K** * (Vx + Vy).

Model	Parameter	Units	Description
Sub	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Subtraction: V = **K** * (Vx - Vy).

Model	Parameter	Units	Description
Max	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Maximum: V = **K** * max(Vx, Vy).

if Vx >= Vy . . : V = **K** * Vx
if Vx < Vy . . . : V = **K** * Vy

Model	Parameter	Units	Description
Min	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Maximum**:** V = **K** * max(Vx, Vy).

if Vx >= Vy . . : V = **K** * Vx
if Vx < Vy . . . : V = **K** * Vy

Model	Parameter	Units	Description
Pwr	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

"Signed" power function**:** V = **K** * pwr(V*x*, V*y*).

The function is calculated as follows:

if V*y* = 0:
- if V*x* < 0. . . : V = **-K**
- if V*x* = 0. . . : V = 0
- if V*x* > 0. . . : V = **K**

if V*y* ≠ 0:
- if V*x* < 0. . . : V = **- K** * $(-Vx)^{Vy}$
- if V*x* = 0. . . : V = 0
- if V*x* > 0. . . : V = **K** * Vx^{Vy}

Model	Parameter	Units	Description
Mag	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Magnitude**.** V = **K** * sqrt($Vx^2 + Vy^2$).

Model	Parameter	Units	Description
Phase	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Phase: V = **K** * phase(V*x*, V*y*).

V in Volts is equal to phase of a vector V*x* + jV*y* in degrees.
If V*x* = 0 and V*y* = 0: V = 0.

Model	Parameter	Units	Description
GT	**IC**	V	Initial condition: output voltage.

Greater than: V = V*x* > V*y* ? **High** : **Low**.

High and **Low** are logical levels.

Model	Parameter	Units	Description
LT	**IC**	V	Initial condition: output voltage.

Less than. V = V*x* < V*y* ? **High** : **Low**.

High and **Low** are logical levels.

Model	Parameter	Units	Description
Table	**X**		Comma-separated string, X (input values).
	Y		Comma-separated string, Y (input values).
	Table		Comma-separated string, Table of Z (output values).
	IC	V	Initial condition: output voltage.

2D look-up table. **Table** parameter defines output Z as a function of X and Y inputs of the component in the following format:

Z11,Z12,…,Z1N,Z21,Z22,…,Z2N,…,ZM1,ZM2,…,ZMN

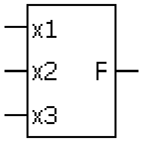
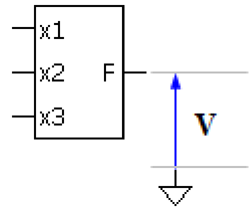
where:

- Zij defines output of the function for input values Xi and Yj;
- N is total number of X input values, defined by **X** parameter;
- M is total number of Y input values, defined by **Y** parameter.

Output value between specified X and Y points is linearly interpolated on both coordinates. Output value below X1 is linearly extrapolated using X1…X2 interval data, output value above XN is linearly extrapolated using X(N-1)…XN interval data. The same rule is applied to Y coordinate

See *Working with 2D Table model* chapter for more details.

F – Custom function

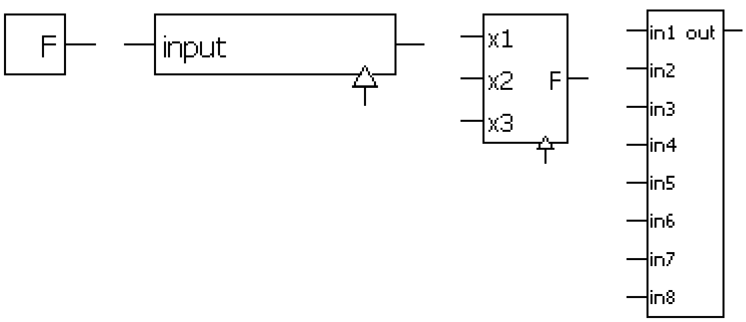
Symbol	Models	Traces
	Function	

This is a customized component. it can be edited n the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 8(height),
- up to 8 inputs on the left side,
- one output on the right side,
- one or no clock pins on the bottom side,
- custom input and output names.

Examples of Custom function component:



Model	Parameter	Units	Description
Function	**f**	V	Output as function of the inputs.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

pin_name – input voltage on the input pin **pin_name**.
t **-** current time
V(*name*) **-** voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

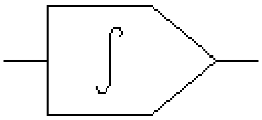
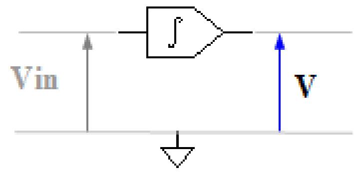
Example:

f = max(x1, x2, x3)
f = (in1+in2) * V(R1)

If ***clock*** pin does not exist, the model operates in **continuous** mode: the function is calculated and applied to the output on every calculation step. If ***clock*** pin exists, the model operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation than **continuous** mode.

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that input voltages and variables **V**, **I, P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

F – Integral

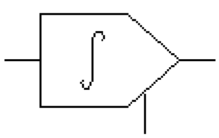
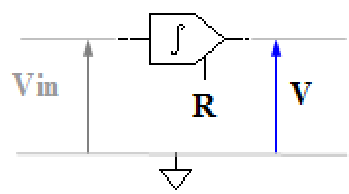
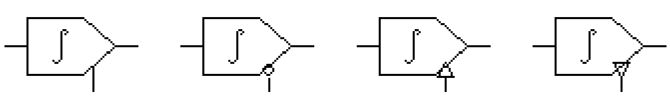
Symbol	Models	Signals
	Integral	

Model	Parameter	Units	Description
Integral	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Integral: V = **K** * $\int$ V*in* dt.

When calculating DC operating point, output is set to specified output voltage **IC**.

F – Integral with reset

Symbol	Models	Signals
	Integral	
Views		

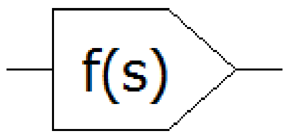
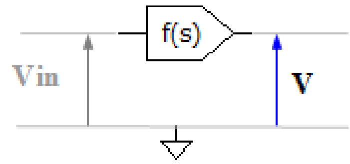
Model	Parameter	Units	Description
Integral	**K**	V/V	Gain.
	IC	V	Initial condition: output voltage.

Integral: V = **K** * $\int$ V*in* dt.

If reset signal **R** is active, output is always zero. If rising or falling edge reset signal applied, output is set to zero and integration continues.

When calculating DC operating point, output is set to specified output voltage **IC**.

F – s-function

Symbol	Models		Signals
	Function Poly1 Poly2 Poly3	Poly4 Poly5 Roots	

Model	Parameter	Units	Description
Function	**f**	V/V	Transfer function.

Transfer function **f** defines transfer function in ***s*** domain. The following variables can represent frequency in the function:

f – current AC frequency, Hz
w – angular AC frequency, ***w = 2πf*** .
s or ***p*** – Laplace parameter, ***s = p = j*2πf***.

Example:

f = 1/(1+s)
f = exp(-R1*C1**s)

Only operators and functions that support complex numbers can be used in this function. If **f** is blank, it is assumed to be **1**.

At transient and DC operation point calculation for AC (if enabled), the component behaves as a buffer with infinite bandwidth, and gain equal to **f(0)**.

Model	Parameter	Units	Description
Poly1 Poly2 Poly3 Poly4 Poly5	**b0**		Numerator polynomial coefficients 0.
	…		**…**
	a0		Denominator polynomial coefficients 0.
	…		**…**
	IC		Initial condition.

AC transfer function is a ratio of polynomials of Laplace parameter s:

f(s) = (**b0** + **b1***s + **b2***s^2 + …) / (**a0** + **a1***s + **a2***s^2 + …)

These models support transient as well.

Initial condition **IC** is a **csv** (comma separated values) string, where the first value is initial output voltage, and other values are internal model values (derivatives of input and output signal).
IC can be modified manually:

- Clear **IC** string and leave it blank to indicate that initial conditions are not specified.
- Enter just one value – initial output voltage.
- Modify the first value – initial output voltage.

If **Save IC** command was performed, then modifying **IC** parameter manually is not recommended.

At DC operation point calculation, **f(0)** is used.

Model	Parameter	Units	Description
Roots	**K**		Gain.
	Roots		Roots (zeroes and poles).
	IC	V	Initial condition.

AC transfer function is defines by zeroes and poles:

f(s) = **K** * (s-z1)*(s-z2)… /(s-p1)/(s-p2)…

where **K** is gain, z1…zn are zeroes, p1…pN are poles.

Roots are defined by **Roots** parameter in the **csv** (comma-separated values) format, as follows:

Nz,Rez1,Imz1,…,Np,Rep1,Imp1,…

where:

Nz - number of zeroes
Rezi – real part of zi
Imzi – imaginary part of zi
Np - number of poles,
Repi – real part of pi
Impi – imaginary part of pi

There could be any number of zeroes and poles, however the resulting numerator and denominator polynomials order should not exceed 5. See *Working with Roots model* chapter for details on entering/editing roots.

The model supports transient as well.

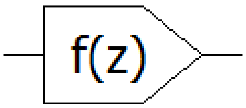
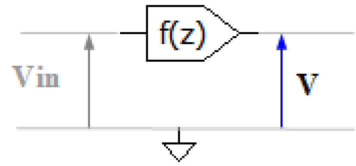
Initial condition **IC** is a **csv** (comma separated values) string, where the first value is initial output voltage, and other values are internal model values (derivatives of input and output signal).
IC can be modified manually:

- Clear **IC** string and leave it blank to indicate that initial conditions are not specified.
- Enter just one value – initial output voltage.
- Modify the first value – initial output voltage.

If **Save IC** command was performed, then modifying **IC** parameter manually is not recommended.

At DC operation point calculation, **f(0)** is used.

F – z-function

Symbol	Models	Signals
	Function Poly1 Poly2 Poly3 Poly4 Poly5	

Model	Parameter	Units	Description
Function	**f**	V/V	Transfer function.

Transfer function **f** defines transfer function in ***z*** domain. The following variables can represent frequency in the function:

> ***f*** – current AC frequency, Hz
> ***w*** – angular AC frequency, ***w = 2πf*** .
> ***s*** or ***p*** – Laplace parameter, ***s = p = j*2πf***.
> ***z*** – z-parameter.

Definition of z-parameter is located in *AC settings/Advanced settings/AC* window. Please note that if T parameter is used in z-parameter formula - for example, exp(s*T) - it should be defined as a schematic variable.

If **f** is blank, it is assumed to be **1**.

At transient and DC operation point calculation for AC (if enabled), the component behaves as a buffer with infinite bandwidth, and gain equal to the transfer function value at zero frequency.

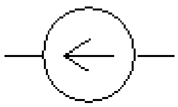
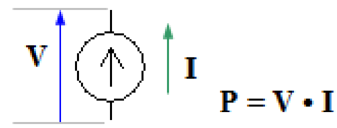
Model	Parameter	Units	Description
Poly1 **Poly2** **Poly3** **Poly4** **Poly5**	**b0**		Numerator polynomial coefficients 0.
	…		…
	a0		Denominator polynomial coefficients 0.
	…		…

AC transfer function is a ratio of polynomials of z-parameter:

$$f(z) = (\mathbf{b0} + \mathbf{b1}{*}z^{-1} + \mathbf{b2}{*}z^{-2} + \ldots) / (\mathbf{a0} + \mathbf{a1}{*}z^{-1} + \mathbf{a2}{*}z^{-2} + \ldots)$$

At transient, and DC operation point calculation for AC (if enabled), the component behaves as a buffer with infinite bandwidth, and gain equal to the transfer function value at zero frequency.

I – Current source

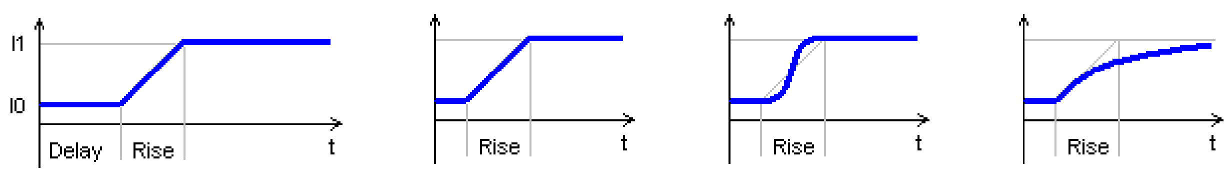
Symbol	Models		Signals
	I Step Single Pulse Clock Sin	Sweep Function List File Trace SubCir	

Model	Parameter	Units	Description
I	**I**	A	Current.

Constant current = **I**.

Model	Parameter	Units	Description
Step	**I1**	A	Step On current.
	I0	A	Step Off current.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

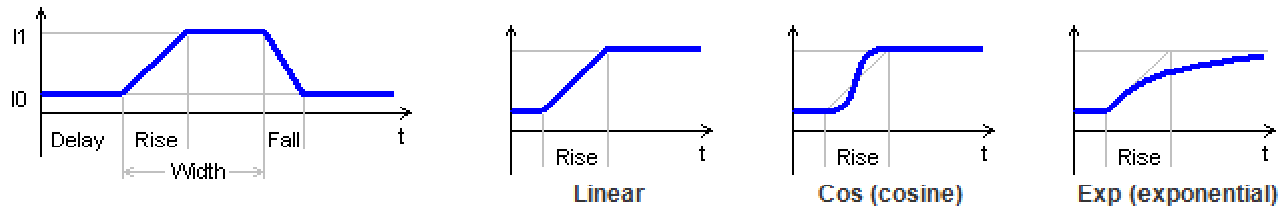
Step starts after **Delay** time. If **Rise** is non-zero, 3 **Slope** types are available.



Linear Cos (cosine) Exp (exponential)

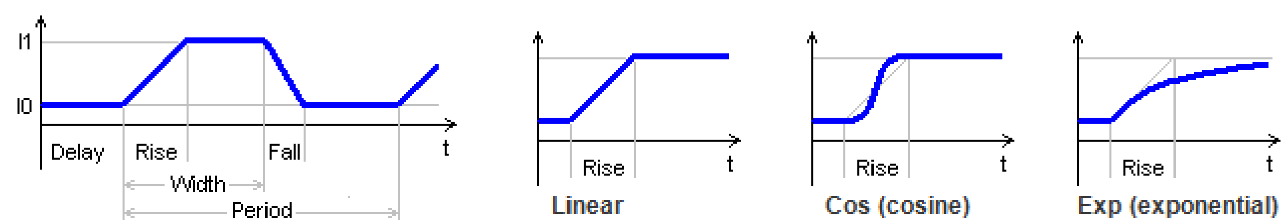
Model	Parameter	Units	Description
Single	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

Single pulse starts after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



Model	Parameter	Units	Description
Pulse	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

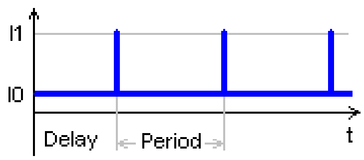
Pulses start after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



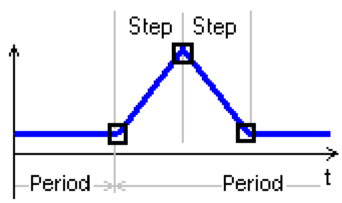
Model	Parameter	Units	Description
Clock	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

Periodic pulses with width of one simulation step. Pulses start after **Delay** time.

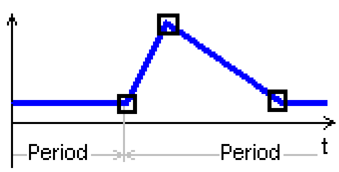
Clock model is recommended to produce a constant frequency clock signal. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:

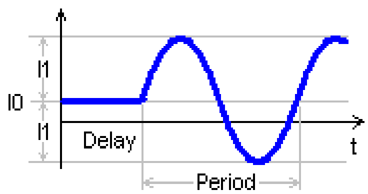


Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:

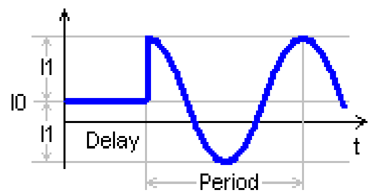


Model	Parameter	Units	Description
Sin	**I1**	A	Current amplitude.
	I0	A	Current baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

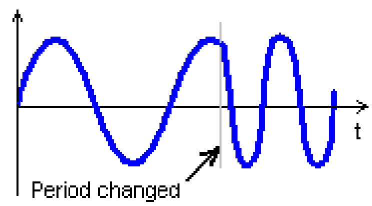
Sine signal starts after **Delay** time. **Phase** is sine phase in degrees at the moment when signal starts. If transient is paused, sine period changed, and then transient is continued, the phase of the signal remains continuous, providing smooth sine signal of variable frequency. If **Decay** is not zero, the sine signal is exponentially dumped with time constant = 1/**Decay**.



Phase = 0



Phase = 90



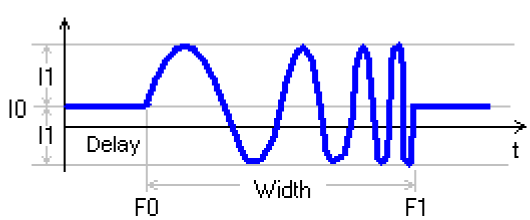
Model	Parameter	Units	Description
Sweep	**I1**	V	Current amplitude.
	I0	V	Current baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

Sinusoidal signal with variable frequency starts after **Delay** time. Signal frequency changes during **Width** interval from **F0** to **F1** linearly or exponentially, depending on specified **Type.**

If **F0** = **F1**, then one period of frequency 1/**Width** will be generated.

If lowest frequency is set to zero and **Type** = Exp, then lowest frequency 0.01/**Width** will be used.

If needed, the highest frequency will be increased to provide integer number of signal periods, so that signal phase at the beginning and at the end of **Width** interval is exactly zero.



Model	Parameter	Units	Description
Function	**f**	A	Function

Arbitrary function **f** defines current as a function of the following variables:

t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of any component in the schematic. If **f** is blank, current is zero.

Example:

f = sin(t) * (1+cos(t*.01))
f = V(R1) * I(R1)

Please note that **V**, **I**, **P**, and **S** variables are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined by **List** parameter in the **csv** (comma-separated values) format, as follows:

t0,I0,t1,I1,...,tn,In

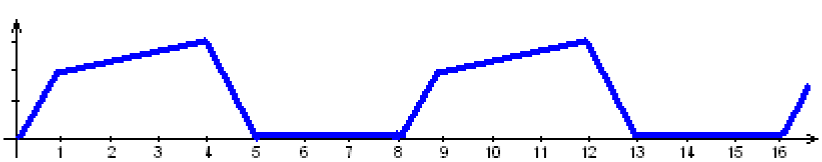
where all t and I can be numerical values or expressions.
If t<t0, signal is I0.
If t0<t<t1, signal value is linearly interpolated between I0 and I1, etc.
If t>tn, and **Cycle** parameter is set to **No**, the signal value is In. Otherwise the signal defined in t0…tn interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
List = 0,0,1,2,4,3,5,0,8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following current will be generated:



See *Working with List source* chapter for more details.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined in the text file.If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Signal is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored>
t0,I0
t1,I1
.....
tn,In
```

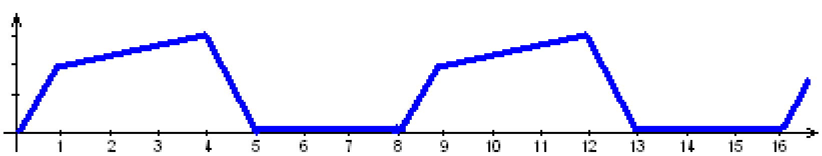
where all t and I can be numerical values or expressions.
If t<t0, signal is v0.
If t0<t<t1, signal value is linearly interpolated between I0 and I1, etc.
If t>tn, and **Cycle** parameter is set to **No**, the signal value is In. Otherwise the signal defined in t0…tn interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
0,0
1,2
4,3
5,0
8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following current will be generated:



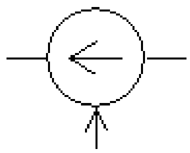
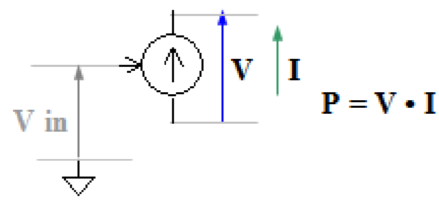
Model	Parameter	Units	Description
Trace	**Trace**		Trace name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Signal is defined by an existing trace. **Trace** parameter is a name of a transient trace. Only traces loaded from data file, imported from text or binary file, duplicated, or pasted from the clipboard can be used.

Signal start is delayed by **Delay** time.

If **Cycle** parameter is set to **Yes**, the signal is repeated continuously.

I – Logic controlled current source

Symbol	Models	Signals
	I One-shot Step Single Pulse Clock Sin Sweep Function List File Trace SubCir	

Model	Parameter	Units	Description
I	**I**	A	Current.
Constant current = **I**.			

Model	Parameter	Units	Description
One-shot	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.

One-shot pulse generator. When increasing input voltage V*in* crosses logical threshold, current pulse of **Width** duration is generated. **I0** is pulse Off level, **I1** is pulse On level.

If increasing V*in* crosses logical threshold value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
Step	**I1**	A	Step On current.
	I0	A	Step Off current.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Step** model of **Current source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Single	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Single** model of **Current source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Pulse	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Pulse** model of **Current source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Clock	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Clock** model of **Current source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Sin	**I1**	A	Current amplitude.
	I0	A	Current baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

When control signal V*in* is below logical threshold, output current is **I0**. When increasing control signal V*in* crosses logical threshold, a signal similar to **Sin** model of **Current source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to **I0** immediately.

Model	Parameter	Units	Description
Sweep	**I1**	V	Current amplitude.
	I0	V	Current baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

When control signal V*in* is below logical threshold, output current is **I0**. When increasing control signal V*in* crosses logical threshold, a signal similar to **Sweep** model of **Current source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to **I0** immediately.

Model	Parameter	Units	Description
Function	**f**	A	Function

When control signal V*in* is below logical threshold, output is zero. When increasing control signal V*in* crosses logical threshold, a signal similar to **Function** model of **Current source** component is generated. If the function is using current time variable ***t***, this moment will be considered as ***t***=0. When decreasing control signal V*in* drops below logical threshold, output goes to zero immediately

Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is equal to **I0** value of **List** signal. When increasing control signal V*in* crosses logical threshold, a signal similar to **List** model of **Current source** component is generated. This moment is also considered as ***t***=0 for the **List** signal. When decreasing control signal V*in* drops below logical threshold, output goes to **I0** immediately.

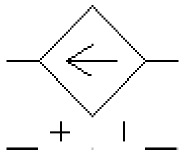
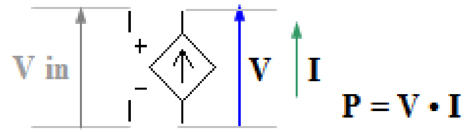
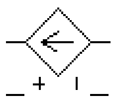
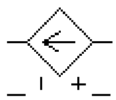
Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is equal to **I0** value specified in the **File**. When increasing control signal V*in* crosses logical threshold, a signal similar to **File** model of **Current source** component is generated. This moment is also considered as ***t***=0 for the **File** signal. When decreasing control signal V*in* drops below logical threshold, output goes to **I0** immediately.

Model	Parameter	Units	Description
Trace	**Trace**		Trace name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is zero. When increasing control signal V*in* crosses logical threshold, a signal similar to **Trace** model of **Current source** component is generated. This moment is also considered as ***t***=0 for the **Trace** signal. When decreasing control signal V*in* drops below logical threshold, output goes to zero immediately.

I – Voltage controlled current source

<table>
<tr><th>Symbol</th><th>Models</th><th>Signals</th></tr>
<tr><td></td><td>Linear
I
Function
PWL
VCO
One-shot
PWM
SubCir</td><td>
</td></tr>
<tr><td>Views</td><td colspan="2"> </td></tr>
</table>

Model	Parameter	Units	Description
Linear	**K**	A/V	Gain

Linear voltage controlled current source: I = **K** * V*in*.

Model	Parameter	Units	Description
I	**I**	A	Current.

Constant current = **I**.

Model	Parameter	Units	Description
Function	**f**	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable ***x*** (input voltage V*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(Vin)

Piecewise linear voltage controlled current source. **pwl** string defines gain as a function of input voltage K(V*in*). The transfer function of the source I(V*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
VCO	**I1**	A	Current amplitude.
	I0	A	Current baseline or Off level.
	dFdV	Hz/V	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Voltage controlled oscillator. Output current is a signal with frequency equal to:

f(Hz) = **dFdV** * Vin.

For **Sine** signal, **I0** is baseline, and **I1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **I0** is Off level, **I1** is On level. **Phase** is additional phase of the signal, in degrees.

Model	Parameter	Units	Description
One-shot	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Threshold	V	Voltage threshold.

One-shot pulse generator. When increasing input voltage V*in* crosses **Threshold** value, current pulse of **Width** duration is generated. **I0** is pulse Off level, **I1** is pulse On level.

If increasing V*in* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
PWM	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	F	Hz	Frequency.
	Vmax	V	Input voltage corresponding to 100% duty.
	Phase	deg	Phase.

Voltage controlled Pulse-Width Modulator. Output current is a pulse signal of frequency **F** shifted by **Phase**. Input voltage V*in* is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:

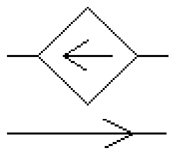
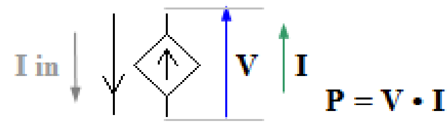
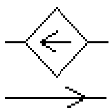
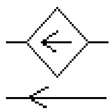
width = 1/**F** * (V*in* / **Vmax**)

or

duty = 100% * (V*in* / **Vmax**);

If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency **F**, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always **F**.

I – Current controlled current source

Symbol	Models		Signals
	Linear I Function PWL	CCO One-shot PWM SubCir	
Views	 		

Model	Parameter	Units	Description
Linear	**K**	A/A	Gain

Linear current controlled current source: I = **K** * I*in*.

Model	Parameter	Units	Description
I	**I**	A	Current.

Constant current = **I**.

Model	Parameter	Units	Description
Function	**f**	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input current I*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable ***x*** (input current I*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(I*in*)

Piecewise linear current controlled current source. **pwl** string defines gain as a function of input current K(I*in*). The transfer function of the source I(I*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
CCO	**I1**	A	Current amplitude.
	I0	A	Current baseline or Off level.
	dFdI	Hz/A	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Current controlled oscillator. Output current is a signal with frequency equal to:

f(Hz) = **dFdI** * Iin.

For **Sine** signal, **I0** is baseline, and **I1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **I0** is Off level, **I1** is On level. **Phase** is additional phase of the signal, in degrees.

Model	Parameter	Units	Description
One-shot	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Threshold	A	Current threshold.

One-shot pulse generator. When increasing input current I*in* crosses **Threshold** value, current pulse of **Width** duration is generated. **I0** is pulse Off level, **I1** is pulse On level.

If increasing I*in* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
PWM	**I1**	A	Pulse On current.
	I0	A	Pulse Off current.
	F	Hz	Frequency.
	Imax	A	Input current corresponding to 100% duty.
	Phase	deg	Phase.

Current controlled Pulse-Width Modulator. Output current is a pulse signal of frequency **F** shifted by **Phase**. Input current I*in* is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:

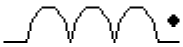
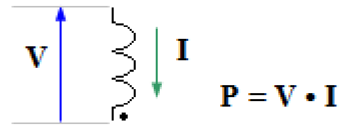
width = 1/**F** * (I*in* / **Imax**)

or

duty = 100% * (I*in* / **Imax**);

If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency **F**, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always **F**.

L – Inductor

Symbol	Models	Signals
	L PWL SubCir	

Model	Parameter	Units	Description
L	**L**	H	Inductance
	IC	A	Initial condition: current. Leave blank if IC not defined.

Linear inductor, V = **L** * dI/dt.

When calculating DC operating point, if **IC** is defined, inductor is replaced with current source equal to **IC**. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.

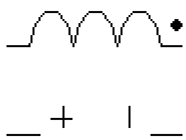
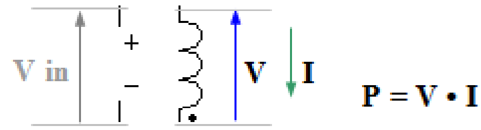
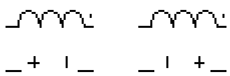
Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, L(I)
	IC	A	Initial condition: current. Leave blank if IC not defined.

Piecewise constant inductor: **pwl** string defines inductance as a function of current through the indictor L(I). H(I) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that L(I) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

When calculating DC operating point, if **IC** is defined, inductor is replaced with current source equal to **IC**. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.

L – Voltage controlled inductor

Symbol	Models	Signals
	PWL	
Views: 		

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, L(V*in*)
	IC	A	Initial condition: current. Leave blank if IC not defined.

Piecewise constant voltage controlled inductor. **pwl** string defines inductance as a function of control voltage L(V*in*). At any moment:

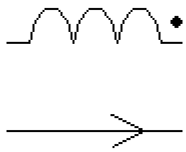
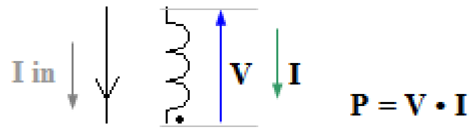
V = L(V*in*) * dI/dt.

See *Working with PWL model* chapter for details.

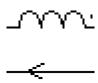
Please note that L(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

When calculating DC operating point, if **IC** is defined, inductor is replaced with current source equal to **IC**. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.

L – Current controlled indictor

Symbol	Models	Signals
	PWL	

Views



Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, L(I*in*)
	IC	A	Initial condition: current. Leave blank if IC not defined.

Piecewise constant current controlled inductor. **pwl** string defines inductance as a function of control current C(I*in*). At any moment:

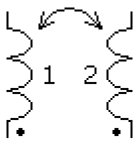
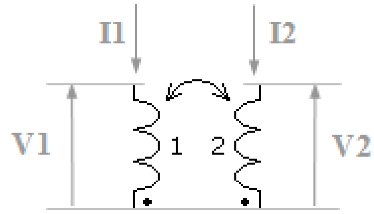
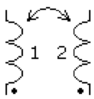
V = L(I*in*) * dI/dt.

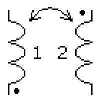
See *Working with PWL model* chapter for details.

Please note that L(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

When calculating DC operating point, if **IC** is defined, inductor is replaced with current source equal to **IC**. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.

L – Coupled inductors

<table>
<tr><th>Symbol</th><th>Models</th><th>Signals</th></tr>
<tr><td>
</td><td>L</td><td>
</td></tr>
<tr><td colspan="3">Views 


</td></tr>
</table>

Model	Parameter	Units	Description
L	**L1**	H	L1 inductance
	L2	H	L2 inductance
	K		Coupling coefficient (-1…1)
	IC1	A	L1 initial condition: current. Leave blank if IC1 not defined.
	IC2	A	L2 initial condition: current. Leave blank if IC2 not defined.

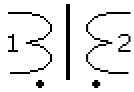
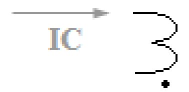
Coupled linear inductors.

V1 = **L1** * dI1/dt + M * dI2/dt
V2 = M * dI1/dt + **L2** * dI2/dt

Where M = **K** * sqrt(**L1** * **L2**) is mutual inductance.

When calculating DC operating point, initial conditions **IC1** and **IC2** are independently applied to corresponding inductors **L1** and **L2**, similar to how it is done for the component **L** (inductor).

L – Custom coupled inductors

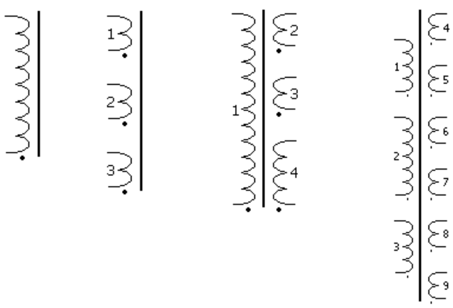
Symbol	Models	Signals
	L SubCir	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- height from 2 to 32,
- up to 32 windings (total),
- arbitrary length of a winding.

Examples of Custom coupled inductors component:



Model	Parameter	Units	Description
L	**L1**	H	L1 inductance
	…	H	**…**
	LN	H	LN inductance
	K12		L1-L2 coupling coefficient (-1…1)
	…		**…**
	K(N-1)N		L(N-1)-LN coupling coefficient (-1…1)
	IC1	A	L1 initial condition: current. Leave blank if IC1 not defined.
	…	A	**…**
	ICN	A	LN initial condition: current. Leave blank if ICN not defined.

Custom coupled inductors.

V1 = **L1** * dI1/dt + M12 * dI2/dt + … + M1N * dIN/dt
V2 = M12 * dI1/dt + **L2** * dI2/dt + … + M2N * dIN/dt
…
VN = M1N * dI1/dt + M2N * dI2/dt + … + **LN** * dIN/dt

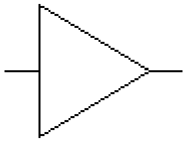
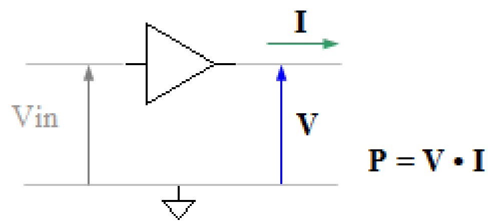
Where Mij = **Kij** * sqrt(**Li** * **Lj**) is mutual inductance, Mij = Mji.
When calculating DC operating point, initial conditions **ICN** are independently applied to corresponding inductors **LN**, similar to how it is done for the component **L** (inductor).

If only one winding is defined, a component behaves exactly as a linear inductor **L**.

Please be aware that coupling coefficients **Kij** should be properly specified within allowable range (-1…1) in order to represent a "physically-realizable" system. If all coupling coefficients are equal to 1 (or -1), using **Winding** components **W** with one magnetizing inductor may give better performance and more stable solution.

If number of windings is more than 9, coupling coefficient parameters **Kij** are shown with underscore '_' between numbers **i** and **j**: for example, **K6_24**. For number of windings 9 and less, the old notation is used for backward compatibility.

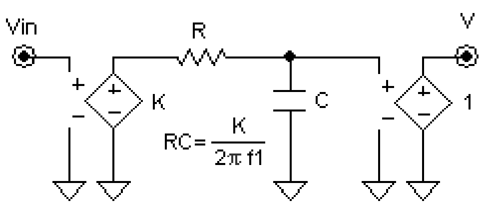
O – Amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

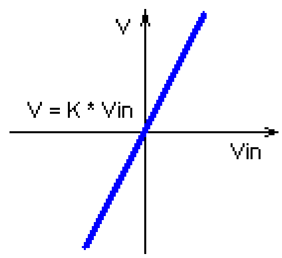
Model	Parameter	Units	Description
Linear	**K**	V/V	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

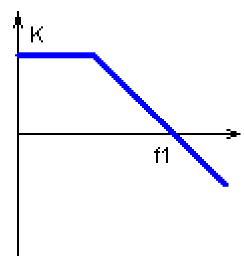
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



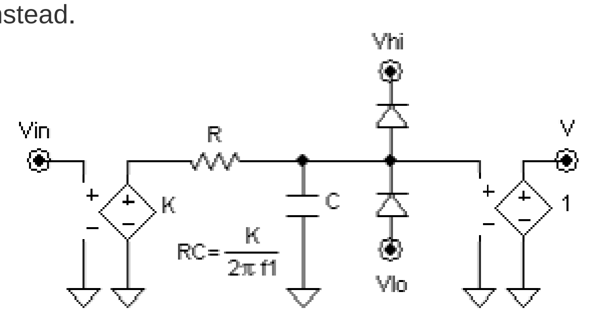
AC response

Model	Parameter	Units	Description
OpAmp	**K**	V/V	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

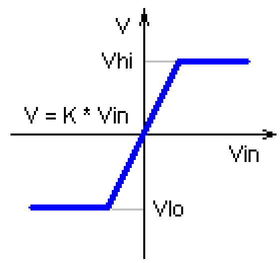
Linear amplifier with output limiter**.** **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.

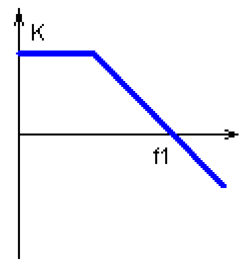
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

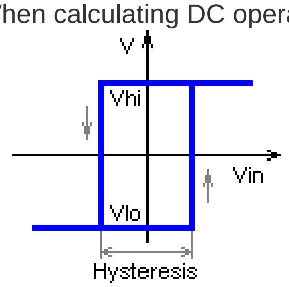
Model	Parameter	Units	Description
Comparator	**Hysteresis**	V	Hysteresis
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Vhi** or **Vlo** using following rules:

V*in* > **Hysteresis**/2 . . : V = **Vhi**
V*in* < **- Hysteresis**/2 . . : V = **Vlo**
Otherwise . . . .: V = previous state

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Vlo** or to **Vhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

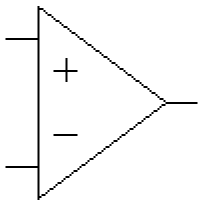
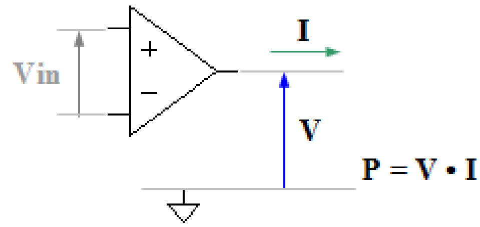
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable ***x*** (input voltage V*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(V*in*)

Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage K(V*in*). Amplifier transfer function is V(V*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

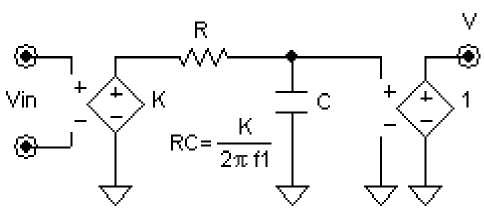
O – Differential amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

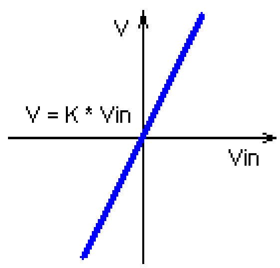
Model	Parameter	Units	Description
Linear	**K**	V/V	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

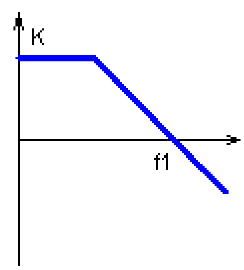
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



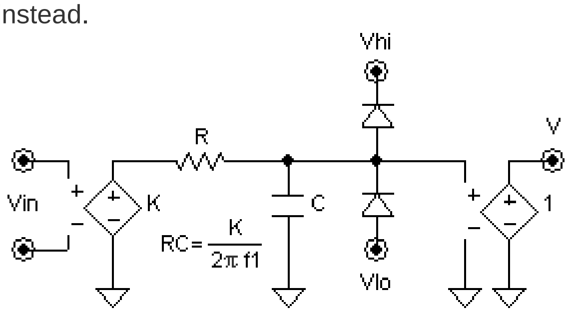
AC response

Model	Parameter	Units	Description
OpAmp	**K**	V/V	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

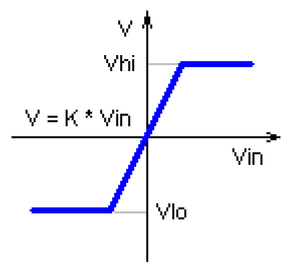
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.

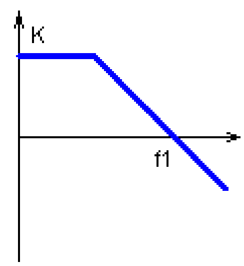
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

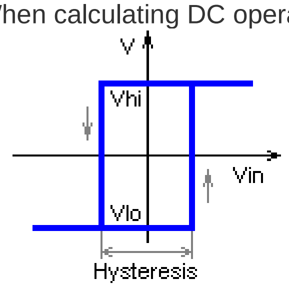
Model	Parameter	Units	Description
Comparator	**Hysteresis**	V	Hysteresis
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Vhi** or **Vlo** using following rules:

V*in* > **Hysteresis**/2 . . : V = **Vhi**
V*in* < - **Hysteresis**/2 . . : V = **Vlo**
Otherwise . . . .: V = previous state

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Vlo** or to **Vhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

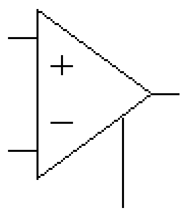
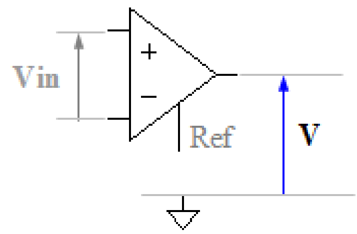
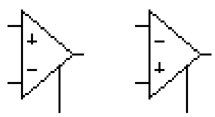
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable ***x*** (input voltage V*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(V*in*)

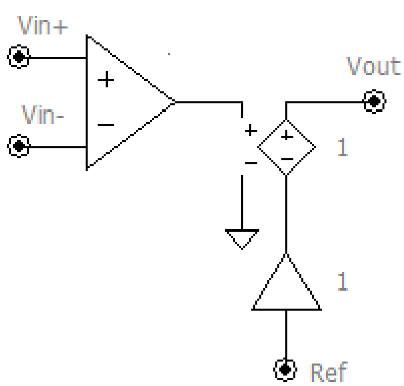
Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage K(V*in*). Amplifier transfer function is V(V*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

O – Differential amplifier with reference

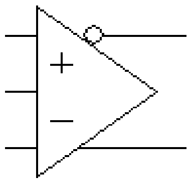
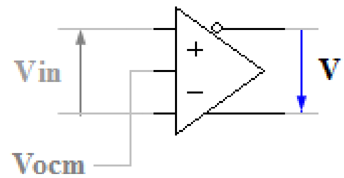
Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	
Views		

Equivalent schematic :

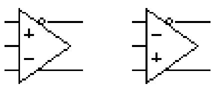


All models are similar to the models of **Differential amplifier** component. Model parameters apply to the differential amplifier shown on the equivalent schematic. For example, **Vhi** and **Vlo** parameters of the **OpAmp** model will limit output of the differential amplifier, rather than voltage at the component output *Vout*.

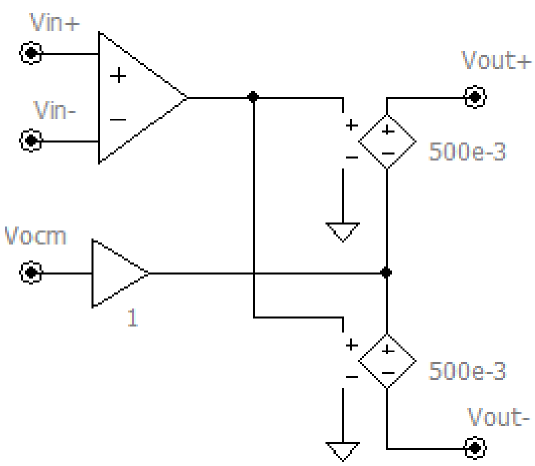
O – Fully differential amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Views

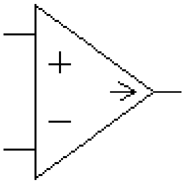
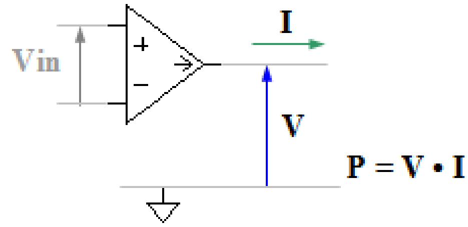


Equivalent schematic:



All models are similar to the models of **Differential amplifier** component. Model parameters apply to the differential amplifier shown on the equivalent schematic. For example, **Vhi** and **Vlo** parameters of the **OpAmp** model will limit output of the differential amplifier, rather than voltages at the component outputs *Vout+* and V*out-*.

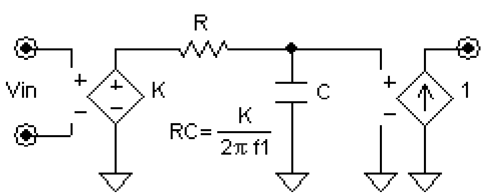
O – Differential amplifier with current output

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

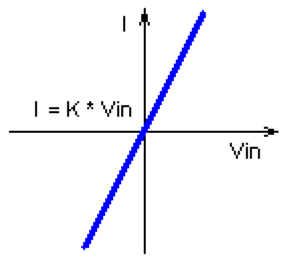
Model	Parameter	Units	Description
Linear	**K**	A/V	Gain
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: output current.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

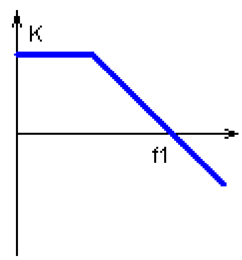
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



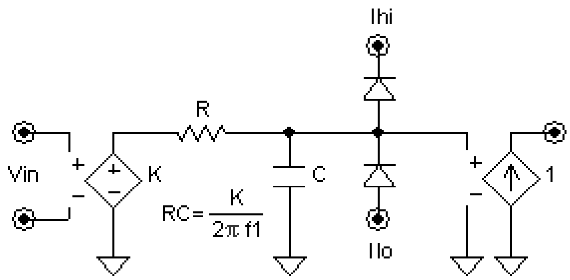
AC response

Model	Parameter	Units	Description
OpAmp	**K**	V/A	Gain
	f1	Hz	Unit gain frequency.
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	IC	A	Initial condition: output current.

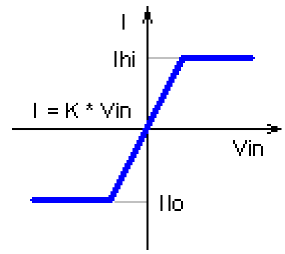
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output current is limiting between **Ilo** and **Ihi**.

When calculating DC operating point, amplifier output is set to specified output current **IC**.

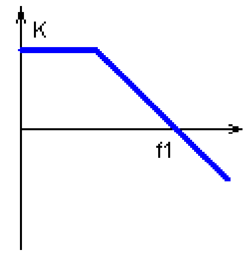
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

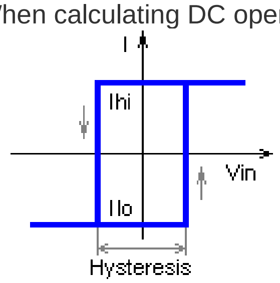
Model	Parameter	Units	Description
Comparator	**Hysteresis**	V	Hysteresis
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Ihi** or **Ilo** using following rules:

V*in* > **Hysteresis**/2 . . : I = **Ihi**
V*in* < - **Hysteresis**/2 . . : I = **Ilo**
Otherwise . . . .: I = previous state

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Ilo** or to **Ihi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	**f**	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

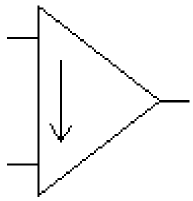
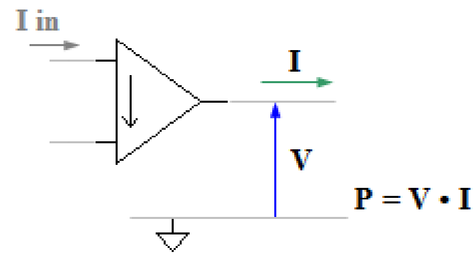
When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable ***x*** (input voltage V*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(V*in*)

Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage K(V*in*). Amplifier transfer function is I(V*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

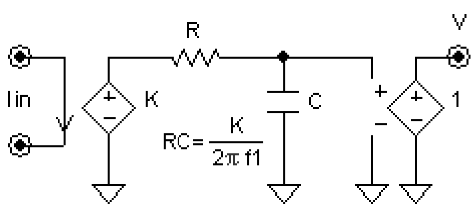
O – Current amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

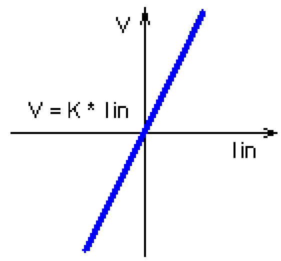
Model	Parameter	Units	Description
Linear	**K**	V/A	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

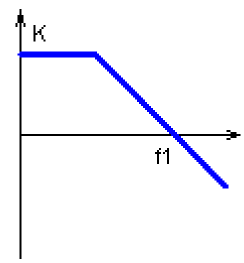
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



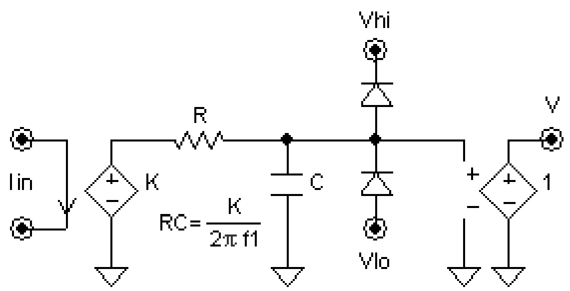
AC response

Model	Parameter	Units	Description
OpAmp	**K**	V/A	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

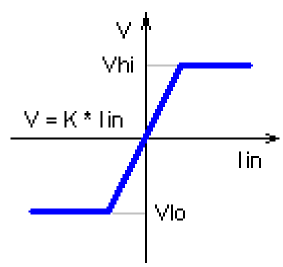
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.

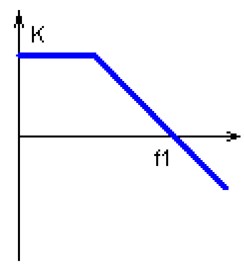
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

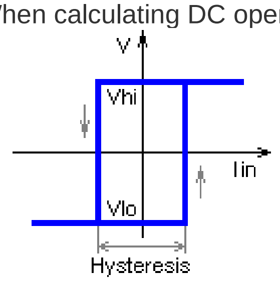
Model	Parameter	Units	Description
Comparator	**Hysteresis**	A	Hysteresis
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Vhi** or **Vlo** using following rules:

I*in* > **Hysteresis**/2 . . : V = **Vhi**
I*in* < - **Hysteresis**/2 . . : V = **Vlo**
Otherwise . . . .: V = previous state

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Vlo** or to **Vhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input current I*in*
t **-** current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

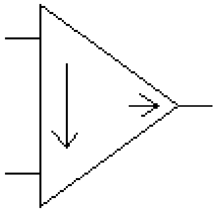
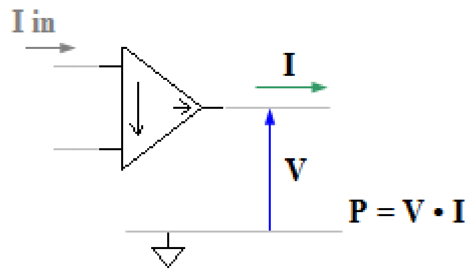
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable ***x*** (input current I*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(V*in*)

Piecewise linear amplifier. **pwl** string defines gain as a function of input current K(I*in*). Amplifier transfer function is V(I*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

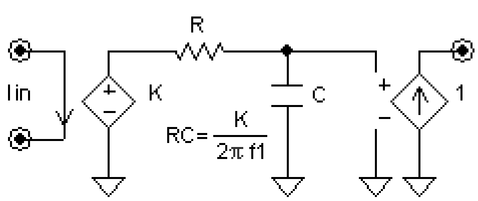
O – Current amplifier with current output

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

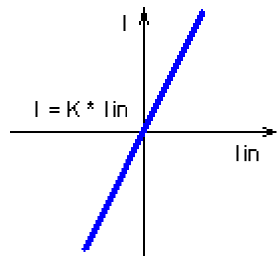
Model	Parameter	Units	Description
Linear	**K**	A/A	Gain
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: output current.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

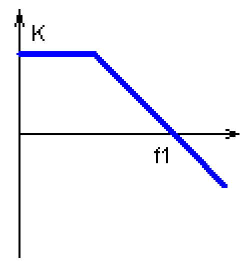
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



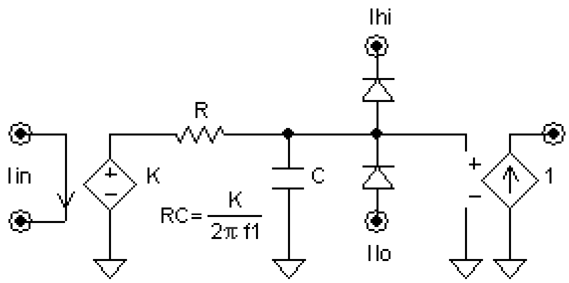
AC response

Model	Parameter	Units	Description
OpAmp	**K**	A/A	Gain
	f1	Hz	Unit gain frequency.
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	IC	A	Initial condition: output current.

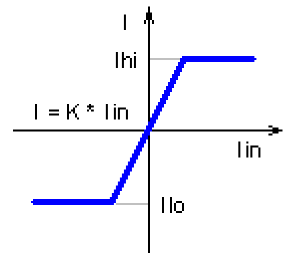
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output current is limiting between **Ilo** and **Ihi**.

When calculating DC operating point, amplifier output is set to specified output current **IC**.

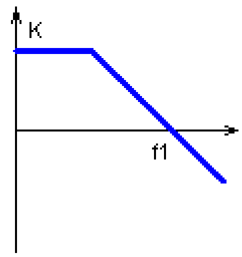
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

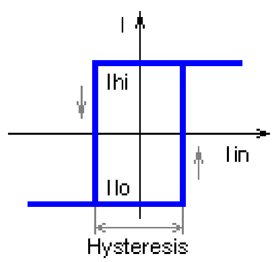
Model	Parameter	Units	Description
Comparator	**Hysteresis**	A	Hysteresis
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Ihi** or **Ilo** using following rules:

I*in* > **Hysteresis**/2 . . : I = **Ihi**
I*in* < - **Hysteresis**/2 . . : I = **Ilo**
Otherwise . . ..: I = previous state

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Ilo** or to **Ihi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	**f**	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input current I*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

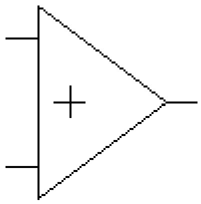
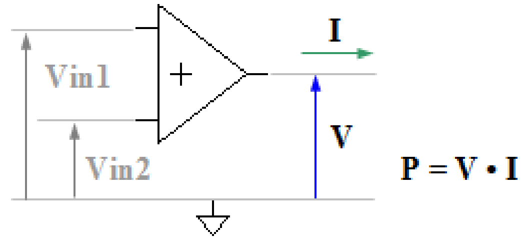
When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable ***x*** (input current I*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(I*in*)

Piecewise linear amplifier. **pwl** string defines gain as a function of input current K(I*in*). Amplifier transfer function is I(I*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

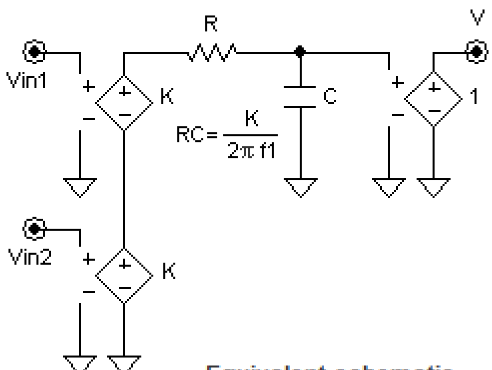
O – Summing amplifier

Symbol	Models	Signals
	Linear OpAmp Function PWL SubCir	

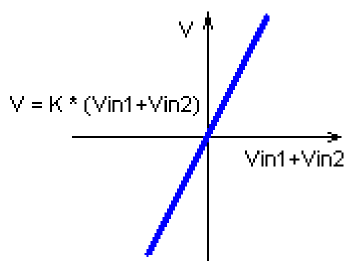
Model	Parameter	Units	Description
Linear	**K**	V/V	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear summing amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

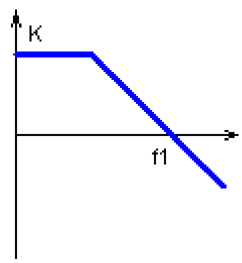
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic

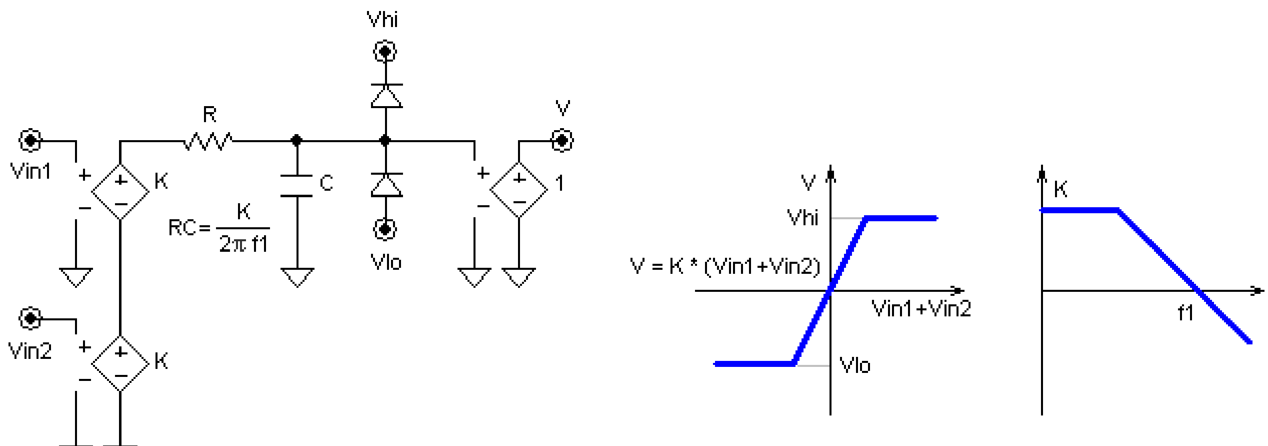


AC response

Model	Parameter	Units	Description
OpAmp	**K**	V/V	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.



Equivalent schematic Static characteristic AC response

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – voltage V*in*1+V*in*2
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

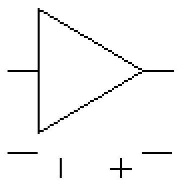
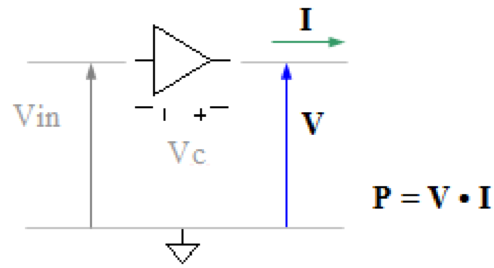
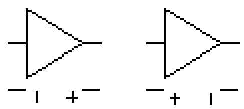
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable ***x*** (input voltage V*in*1+V*in*2) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(V*in*1+V*in*2)

Piecewise linear amplifier. **pwl** string defines gain as a function of sum of input voltages K(V*in*1+V*in*2). Amplifier transfer function is V(V*in*1+V*in*2) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(V*in*1+V*in*2) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

O – Voltage controlled amplifier

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(V*c*)

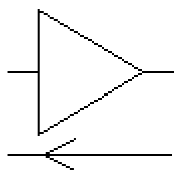
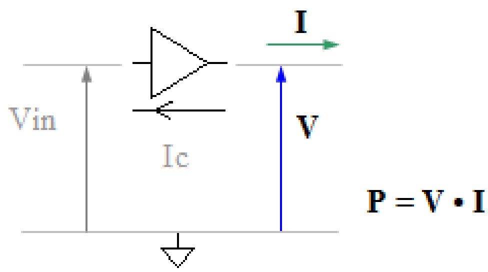
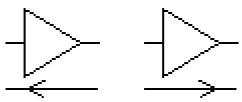
Piecewise constant voltage controlled amplifier. **pwl** string defines gain as a function of control voltage K(V*c*). At any moment:

V = K(V*c*) * V*in*.

See *Working with PWL model* chapter for details.

Please note that K(V*c*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

O – Current controlled amplifier

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(I*c*)

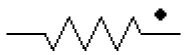
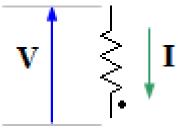
Piecewise constant current controlled amplifier. **pwl** string defines gain as a function of control current K(I*c*). At any moment:

V = K(I*c*) * V*in*.

See *Working with PWL model* chapter for details.

Please note that K(I*c*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

R – Resistor

Symbol	Models	Signals
	R PWL PWL-I SubCir	 **P = V • I**

Model	Parameter	Units	Description
R	**R**	Ohm	Resistance

Linear resistor: V = **R** * I.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, R(V)

Piecewise constant resistor. **pwl** string defines resistance as a function of voltage across the resistor R(V). Current through the resistor I(V) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

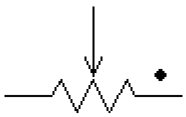
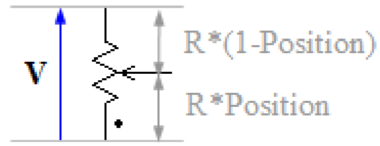
Please note that R(V) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
PWL-I	**pwl**		Comma-separated string, R(I)

Piecewise constant resistor. **pwl** string defines resistance as a function of current through the resistor R(I). Voltage across the resistor V(I) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that R(I) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

R – Potentiometer

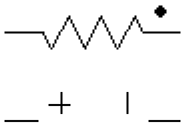
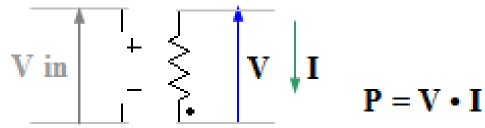
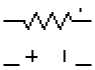
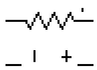
Symbol	Models	Signals
	Potentiometer	

Model	Parameter	Units	Description
Potentiometer	**R**	Ohm	Resistance
	Position		Position of the wiper (0…1)

Position of the wiper is referenced to the terminal with dot:

0 – wiper is connected to the terminal with dot
1 – wiper is connected to another terminal.

R – Voltage controlled resistor

Symbol	Models	Signals
	PWL	
Views	 	

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, R(V*in*)

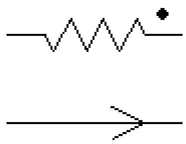
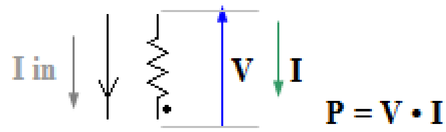
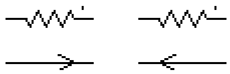
Piecewise constant voltage controlled resistor. **pwl** string defines resistance as a function of control voltage R(V*in*). At any moment:

V = R(V*in*) * I.

See *Working with PWL model* chapter for details.

Please note that R(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

R – Current controlled resistor

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, R(I*in*)

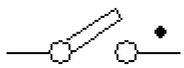
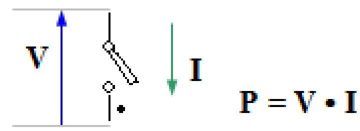
Piecewise constant current controlled resistor. **pwl** string defines resistance as a function of control current R(I*in*). At any moment:

V = R(I*in*) * I.

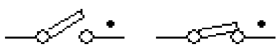
See *Working with PWL model* chapter for details.

Please note that R(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

S – Switch

Symbol	Models	Signals
	Off On Step Single Pulse Clock List File SubCir	

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



Model	Parameter	Units	Description
Off	No parameters.		

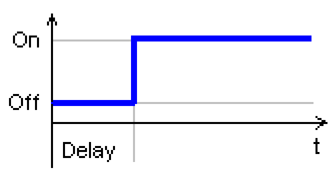
Switch is always in Off state (open circuit).

Model	Parameter	Units	Description
On	No parameters.		

Switch is always in On state (short circuit).

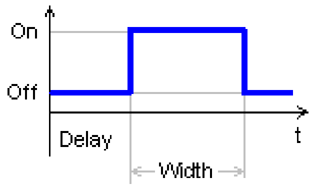
Model	Parameter	Units	Description
Step	**Delay**	s	Delay before active state.
	Active		Active switch state: Off/On.

Switch goes to **Active** state after **Delay** time. Switching diagram for **Active** = On:



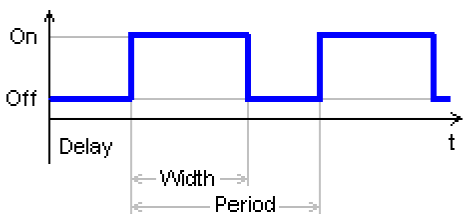
Model	Parameter	Units	Description
Single	**Width**	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

Single pulse starts after **Delay** time. Switching diagram is shown for **Active** = On:



Model	Parameter	Units	Description
Pulse	**Period**	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

Pulses start after **Delay** time. Switching diagram is shown for **Active** = On:

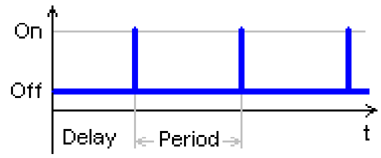


Model	Parameter	Units	Description
Clock	**Period**	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

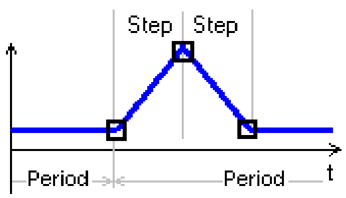
Periodic pulses with width of one simulation step. Pulses start after **Delay** time. Switch goes to **Active** state for one simulation step only.

Unlike **Pulse** model, **Clock** model won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.

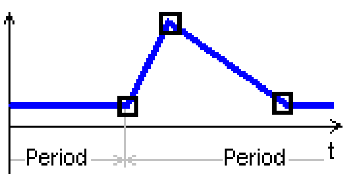
Switching diagram is shown for **Active** = On:



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:



Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:



Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Switching sequence is defined in the **List** parameter in the **csv** (comma-separated values) format, as follows:

t0,s0,t1,s1,...,tn,sn

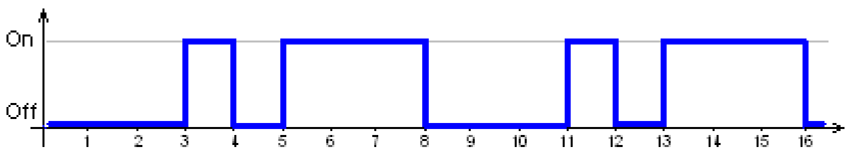
s0…sn defines switch state: positive number corresponds to On state, zero or negative number - Off state.
If t<t0, switch is in s0 state.
At t0 switch is set to s0 state, at t1 switch is set to s1 state, and so on.
If t>tn, and **Cycle** parameter is set to **No**, switch remains in state sn. Otherwise the sequence is repeated continuously.

Switching start is delayed by **Delay** time.

Example:

```
List = 0,0,3,1,4,0,5,1,8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following sequence will be generated:



See *Working with List model* chapter for more details.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Switching sequence defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Switching sequence is defined in the **csv** (comma-separated values) format, as follows:

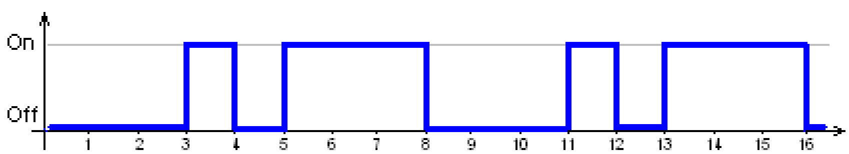
```
<if first line does not start with a number, it is ignored>
t0,s0
t1,s1
.....
tn,sn
```

s0…sn defines switch state: positive number corresponds to On state, zero or negative number - Off state.
If t<t0, switch is in s0 state.
At t0 switch is set to s0 state, at t1 switch is set to s1 state, and so on.
If t>tn, and **Cycle** parameter is set to **No**, switch remains in state sn. Otherwise the sequence is repeated continuously.
Switching start delayed by **Delay** time.

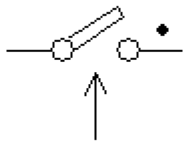
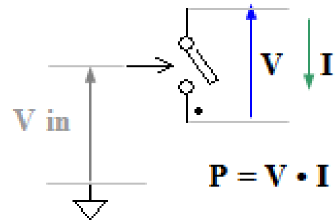
Example:

```
0,0
3,1
4,0
5,1
8,0
```

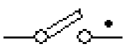
If **Cycle** = **Yes**, **Delay** = 0, the following sequence will be generated:



S – Logic controlled switch

Symbol	Models		Signals
	Switch Off On One-shot Step Single	Pulse Clock List File Steps SubCir	

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:

Model	Parameter	Units	Description
Switch	**Active**		Active state: Off/On.
	IC		Initial condition: Off/On.

Logic controlled switch. Switch is set to active or non-active state using following rules:

V*in* > logical threshold . . . : active
V*in* < logical threshold . . . : non-active

When calculating DC operating point, switch is set to the state defined in **IC**.

Model	Parameter	Units	Description
Off	No parameters.		

Switch is always in Off state (open circuit).

Model	Parameter	Units	Description
On	No parameters.		

Switch is always in On state (short circuit).

Model	Parameter	Units	Description
One-shot	**Width**	s	Pulse width.
	Active		Active state: Off/On.

One-shot switch. When increasing control signal V*in* crosses logical threshold, switch is set to **Active** state for **Width** time interval.

If increasing V*in* crosses logical threshold value while switch is in active state, the **Active** interval is restarted.

Model	Parameter	Units	Description
Step	**Delay**	s	Delay before active state.
	Active		Active switch state: Off/On.

When control signal V*in* is below logical threshold, switch is in non-active state. When increasing control signal V*in* crosses logical threshold, a switching sequence similar to **Step** model of **Switch** component is generated. When decreasing control signal V*in* drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Single	**Width**	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

When control signal V*in* is below logical threshold, switch is in non-active state. When increasing control signal V*in* crosses logical threshold, a switching sequence similar to **Single** model of **Switch** component is generated. When decreasing control signal V*in* drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Pulse	**Period**	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

When control signal V*in* is below logical threshold, switch is in non-active state. When increasing control signal V*in* crosses logical threshold, a switching sequence similar to **Pulse** model of **Switch** component is generated. When decreasing control signal V*in* drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Clock	**Period**	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

When control signal V*in* is below logical threshold, switch is in non-active state. When increasing control signal V*in* crosses logical threshold, a switching sequence similar to **Pulse** model of **Switch** component is generated. When decreasing control signal V*in* drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, switch is in **s0** state defined in the **List** parameter. When increasing control signal V*in* crosses logical threshold, a switching sequence similar to **List** model of **Switch** component is generated. When decreasing control signal V*in* drops below logical threshold, switch goes to **s0** state immediately.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

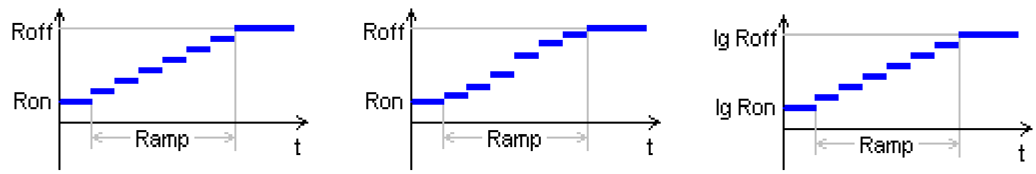
When control signal V*in* is below logical threshold, switch is in **s0** state specified in the **File**. When increasing control signal V*in* crosses logical threshold, a switching sequence similar to **File** model of **Switch** component is generated. When decreasing control signal V*in* drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Steps	**Roff**	Ohm	Off state resistance.
	Ron	Ohm	On state resistance.
	Slope		Type of resistance change: Linear/Cos/Log.
	Ramp	s	Resistance ramp time.
	Steps		Number of resistance steps in the ramp.
	IC		Initial condition: Off/On.

Switch with resistance ramping. When increasing input voltage V*in* crosses logical threshold, switch resistance starts ramping from **Roff** to **Ron**. When decreasing input voltage V*in* crosses logical threshold, switch resistance starts ramping from **Ron** to **Roff**.

Resistance is changing during **Ramp** time interval, with number of steps specified by **Steps** parameter. If **Steps** = 0, resistance is changed instantly.

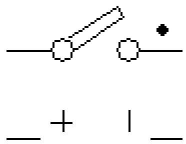
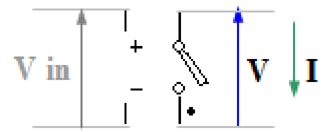
Slope parameter specifies how resistance is changing during the ramp:



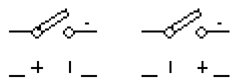
Linear Cos Log

When calculating DC operating point switch is set to the state specified in **IC**.

S – Voltage controlled switch

Symbol	Models		Signals
	Switch Off On	One-shot Steps SubCir	 P = V • I

Views



For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:

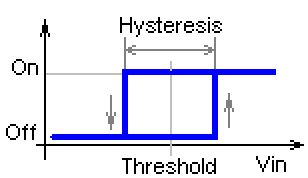
Model	Parameter	Units	Description
Switch	**Threshold**	V	Voltage threshold.
	Hysteresis	V	Hysteresis.
	Active		Active state: Off/On.
	IC		Initial condition: Off/On.

Voltage controlled switch. Switch is set to active or non-active state using following rules:

Vin > **Threshold** + **Hysteresis**/2. . . : active
Vin < **Threshold** - **Hysteresis**/2 . . . : non-active
Otherwise . . . . : previous state

When calculating DC operating point switch is set to the state defined in **IC**.

Switching diagram for **Active** = On:



Model	Parameter	Units	Description
Off	No parameters.		

Switch is always in Off state (open circuit).

Model	Parameter	Units	Description
On	No parameters.		

Switch is always in On state (short circuit).

Model	Parameter	Units	Description
One-shot	**Width**	s	Pulse width.
	Threshold	V	Voltage threshold.
	Active		Active state: Off/On.

One-shot pulse generator. When increasing input voltage V*in* crosses **Threshold** value, switch is set to **Active** state for **Width** time interval.

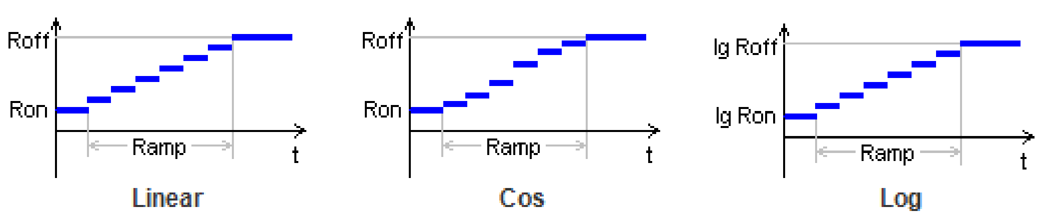
If increasing V*in* crosses logical threshold value while switch is in active state, the **Active** interval is restarted.

Model	Parameter	Units	Description
Steps	**Threshold**	V	Voltage threshold.
	Hysteresis	V	Hysteresis.
	Roff	Ohm	Off state resistance.
	Ron	Ohm	On state resistance.
	Slope		Type of resistance change: Linear/Cos/Log.
	Ramp	s	Resistance ramp time.
	Steps		Number of resistance steps in the ramp.
	IC		Initial condition: Off/On.

Switch with resistance ramping. When increasing input voltage V*in* crosses **Threshold + Hysteresis**/2 value, switch resistance starts ramping from **Roff** to **Ron**. When decreasing input voltage V*in* crosses **Threshold - Hysteresis**/2 value, switch resistance starts ramping from **Ron** to **Roff**.

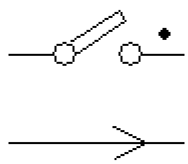
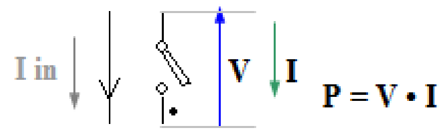
Resistance is changing during **Ramp** time interval, with number of steps specified by **Steps** parameter. If **Steps** = 0, resistance is changed instantly.

Slope parameter specifies how resistance is changing during the ramp:

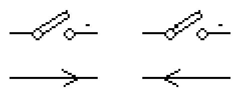


When calculating DC operating point switch is set to the state specified in **IC**.

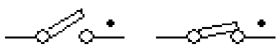
S – Current controlled switch

Symbol	Models		Signals
	Switch Off On	One-shot Steps SubCir	

Views



For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



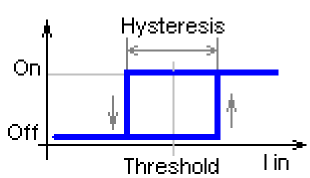
Model	Parameter	Units	Description
Switch	**Threshold**	A	Current threshold.
	Hysteresis	A	Hysteresis.
	Active		Active state: Off/On.
	IC		Initial condition: Off/On.

Current controlled switch. Switch is set to active or non-active state using following rules:

I*in* > **Threshold** + **Hysteresis**/2. . . : active
I*in* < **Threshold** - **Hysteresis**/2 . . . : non-active
Otherwise . . . . : previous state

When calculating DC operating point switch is set to the state defined in **IC**.

Switching diagram for **Active** = On:



Model	Parameter	Units	Description
Off	No parameters.		

Switch is always in Off state (open circuit).

Model	Parameter	Units	Description
On	No parameters.		

Switch is always in On state (short circuit).

Model	Parameter	Units	Description
One-shot	**Width**	s	Pulse width.
	Threshold	A	Current threshold.
	Active		Active state: Off/On.

One-shot pulse generator. When increasing input current I*in* crosses **Threshold** value, switch is set to **Active** state for **Width** time interval.

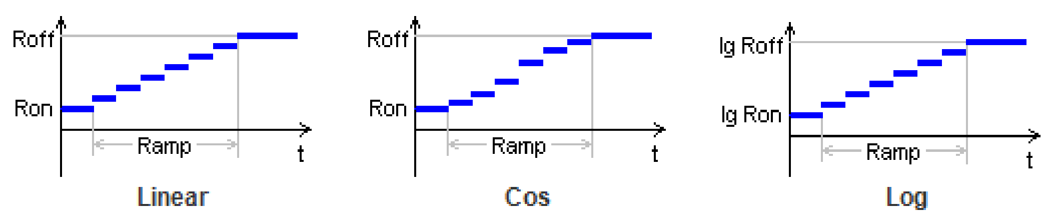
If increasing I*in* crosses logical threshold value while switch is in active state, the **Active** interval is restarted.

Model	Parameter	Units	Description
Steps	**Threshold**	A	Current threshold.
	Hysteresis	A	Hysteresis.
	Roff	Ohm	Off state resistance.
	Ron	Ohm	On state resistance.
	Slope		Type of resistance change: Linear/Cos/Log.
	Ramp	s	Resistance ramp time.
	Steps		Number of resistance steps in the ramp.
	IC		Initial condition: Off/On.

Switch with resistance ramping. When increasing input current I*in* crosses **Threshold + Hysteresis**/2 value, switch resistance starts ramping from **Roff** to **Ron**. When decreasing input current Iin crosses **Threshold - Hysteresis**/2 value, switch resistance starts ramping from **Ron** to **Roff**.

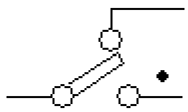
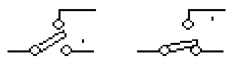
Resistance is changing during **Ramp** time interval, with number of steps specified by **Steps** parameter. If **Steps** = 0, resistance is changed instantly.

Slope parameter specifies how resistance is changing during the ramp:



When calculating DC operating point switch is set to the state specified in **IC**.

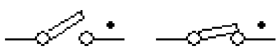
S – SPDT switch

Symbol	Models	Signals
	Off On Step Single Pulse Clock List File SubCir	
Views		

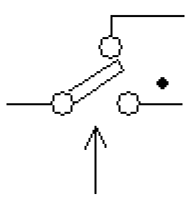
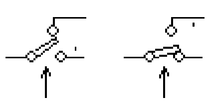
All models and operation of the SPDT (single pole, double throw) switch are similar to single pole **Switch**, with Off and On states defined as follows:

Off state: "common to pin with dot" - open, "common to another pin" - short.
On state: "common to pin with dot" - short, "common to another pin" - open.

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



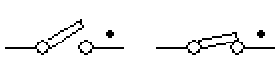
S – SPDT logic controlled switch

Symbol	Models	Signals
	Switch Off On One-shot Step Single Pulse Clock List File Steps SubCir	
Views		

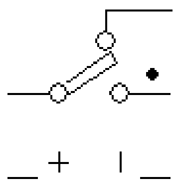
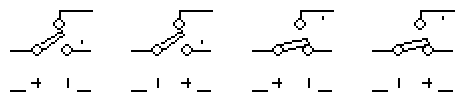
All models and operation of the SPDT (single pole, double throw) logic controlled switch are similar to single pole **Logic controlled switch**, with Off and On states defined as follows:

Off state: "common to pin with dot" - open, "common to another pin" - short.
On state: "common to pin with dot" - short, "common to another pin" - open.

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



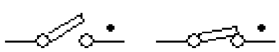
S – SPDT voltage controlled switch

Symbol	Models	Signals
	Switch Off On One-shot Steps SubCir	
Views		

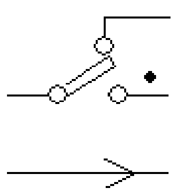
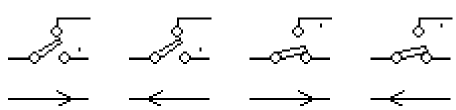
All models and operation of the SPDT (single pole, double throw) voltage controlled switch are similar to single pole **Voltage controlled switch**, with Off and On states defined as follows:

Off state: "common to pin with dot" - open, "common to another pin" - short.
On state: "common to pin with dot" - short, "common to another pin" - open.

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



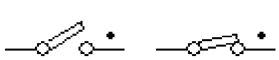
S – SPDT current controlled switch

Symbol	Models	Signals
	Switch Off On One-shot Steps SubCir	
Views		

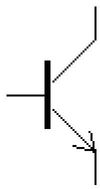
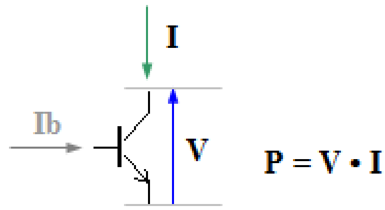
All models and operation of SPDT (single pole, double throw) current controlled switch are similar to single pole **Current controlled switch** component, with Off and On states defined as follows:

Off state: "common to pin with dot" - open, "common to another pin" - short.
On state: "common to pin with dot" - short, "common to another pin" - open.

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



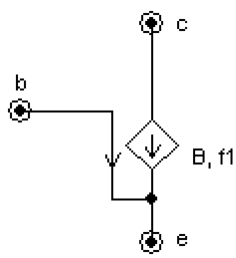
T – NPN transistor

Symbol	Models	Signals
	Linear Switch Transistor SubCir	

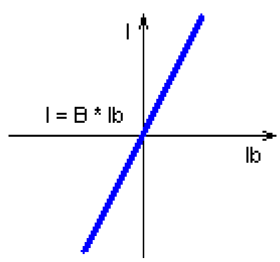
Model	Parameter	Units	Description
Linear	**B**	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: collector current.

Linear BJT transistor. Current controlled current source with specified bandwidth. **B** is open loop gain (beta). Frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

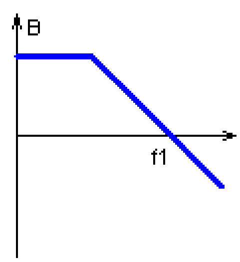
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic

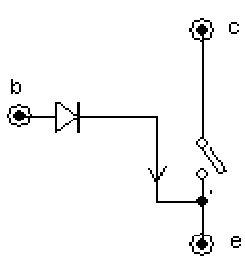


AC response

Model	Parameter	Units	Description
Switch	**Vbe**	V	Forward voltage drop of base-emitter diode.
	IC		Initial condition of base-emitter diode: Off/On.

BJT transistor switch. Current controlled switch with a base-emitter diode. Switch is closed if diode current is non-zero.

When calculating DC operating point, the diode is set to the state specified in **IC**.



Equivalent schematic

Model	Parameter	Units	Description
Transistor	**B**	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	Vbe	V	Forward voltage drop of base-emitter diode.
	Vsat	V	Collector-emitter saturation voltage drop.
	IC	A	Initial condition: collector current.
	ICbe		Initial condition of base-emitter diode: Off/On.
	ICbc		Initial condition of base-collector diode: Off/On.

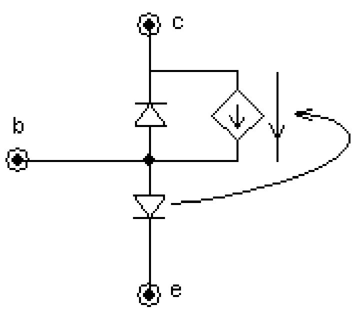
BJT transistor. Simplified Ebers-Moll BJT transistor model with saturation. It consists of two diodes (base-emitter and base-collector), and current source controlled by current through base-emitter diode with gain "alpha":

$$\alpha = \frac{\beta}{1+\beta}$$

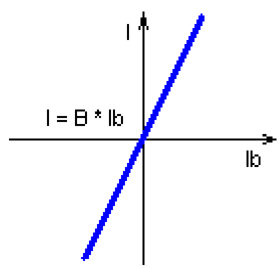
If collector-emitter voltage is higher than **Vsat**, base-collector diode is open, transistor is not saturated, and behaves as **Linear** model (current controlled current source with specified bandwidth). **B** is open loop gain (beta). Low signal frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

If collector voltage drops below **Vsat**, base-collector diode is closed, and transistor is saturated: collector-emitter voltage is equal to **Vsat**.

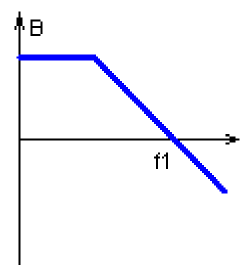
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC.** If **IC** is blank, static characteristic is used. Base-emitter diode is set to the state specified in **ICbe**, Base-collector diode is set to the state specified in **ICbc**.



Equivalent schematic

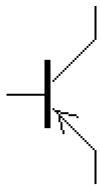
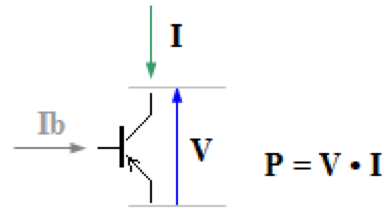


Non-saturated static characteristic



Low signal AC response

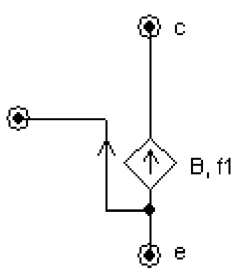
T – PNP transistor

Symbol	Models	Signals
	Linear Switch Transistor SubCir	

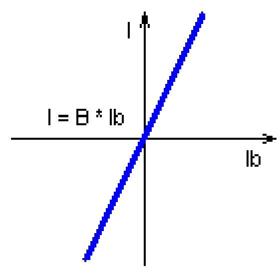
Model	Parameter	Units	Description
Linear	**B**	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: collector current.

Linear BJT transistor. Current controlled current source with specified bandwidth. **B** is open loop gain (beta). Frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

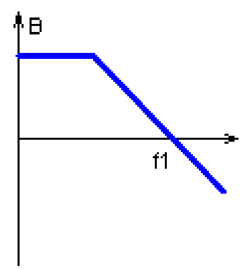
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic

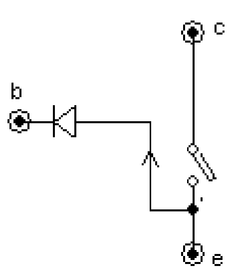


AC response

Model	Parameter	Units	Description
Switch	**Vbe**	V	Forward voltage drop of base-emitter diode.
	IC		Initial condition of base-emitter diode: Off/On.

BJT transistor switch. Current controlled switch with a base-emitter diode.
Switch is closed if diode current is non-zero.

When calculating DC operating point, the diode is set to the state specified in **IC**.



Equivalent schematic

Model	Parameter	Units	Description
Transistor	**B**	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	Vbe	V	Forward voltage drop of base-emitter diode.
	Vsat	V	Collector-emitter saturation voltage drop.
	IC	A	Initial condition: collector current.
	ICbe		Initial condition of base-emitter diode: Off/On.
	ICbc		Initial condition of base-collector diode: Off/On.

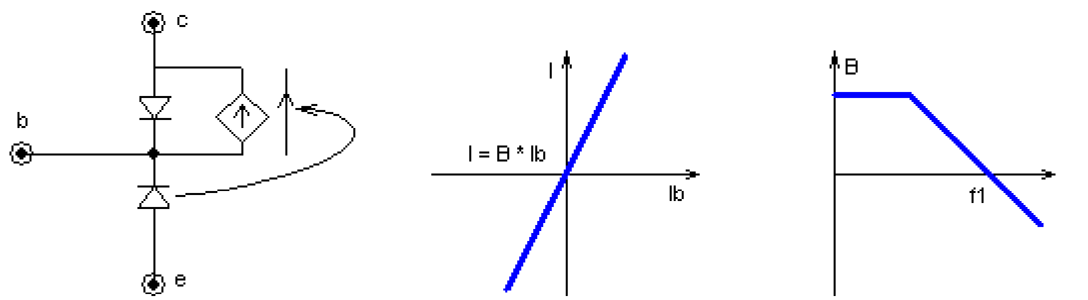
BJT transistor. Simplified Ebers-Moll BJT transistor model with saturation. It consists of two diodes (base-emitter and base-collector), and current source controlled by current through base-emitter diode with gain "alpha":

$$\alpha = \frac{\beta}{1+\beta}$$

If collector-emitter voltage is negative, and less than **-Vsat**, base-collector diode is open, transistor is not saturated, and behaves as **Linear** model (current controlled current source with specified bandwidth). **B** is open loop gain (beta). Low signal frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

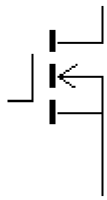
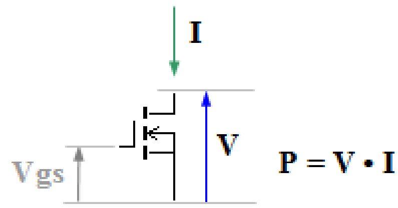
If collector voltage is higher than **-Vsat**, base-collector diode is closed, and transistor is saturated: collector-emitter voltage is equal to **-Vsat**.

When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC.** If **IC** is blank, static characteristic is used. Base-emitter diode is set to the state specified in **ICbe**, Base-collector diode is set to the state specified in **ICbc**.



Equivalent schematic   Non-saturated static characteristic   Low signal AC response

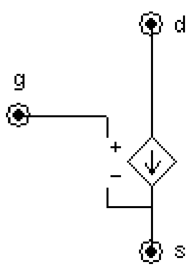
T – N-FET

Symbol	Models	Signals
	Linear Switch FET FET-D SubCir	

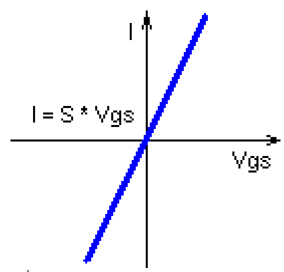
Model	Parameter	Units	Description
Linear	**S**	A/V	Slope
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: drain current.

Linear FET transistor. Voltage controlled current source with specified bandwidth. **S** is open loop slope. Frequency response consists of one pole, **f1** is unit gain frequency. **S** and **f1** can be set to infinity (**inf**).

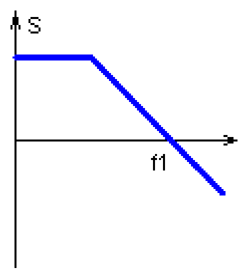
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, drain current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



AC response

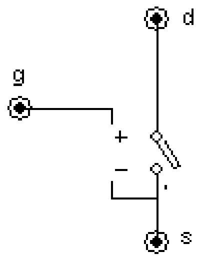
This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

Model	Parameter	Units	Description
Switch	**Vth**	V	Threshold.
	IC		Initial condition of the switch: Off/On.

FET switch. Voltage controlled switch. Switch is closed if gate-source voltage exceeds threshold **Vth**.

When calculating DC operating point switch is set to the state specified in **IC**.

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.



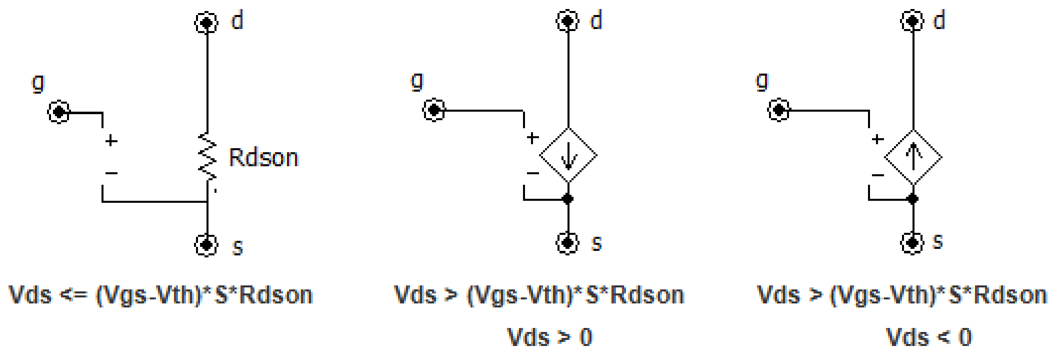
Equivalent schematic

Model	Parameter	Units	Description
FET	**S**	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	IC		Initial condition: Off/R/Plus/Minus

FET transistor. The model has 3 modes of operation.

1. V*gs* <= **Vth** . . . .: I = 0 (open)
2. V*gs* > **Vth**, V*ds* <= (V*gs* - **Vth**) * **S** * **Rdson** . .: V = I * **Rdson** (resistor)
3. V*gs* > **Vth**, V*ds* > (V*gs* - **Vth**) * **S** * **Rdson** . .: I = (V*gs* - **Vth**) * **S** (current source)

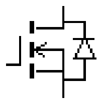
FET works similar for positive and negative drain-source voltage, current direction changes accordingly.
Equivalent schematics (V*gs* > **Vth**):



When calculating DC operating point, transistor is set to an initial state specified by **IC** parameter as follows:

Off . . . : I = 0 (open)
R . . . . : V = I * **Rdson** (resistor)
Plus . . : V*ds* > 0, I = (V*gs* - **Vth**) * **S** ("positive" current source)
Minus. : V*ds* < 0, I = (V*gs* - **Vth**) * **S** ("negative" current source)

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

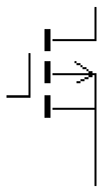
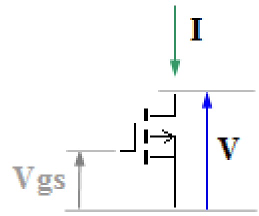
Model	Parameter	Units	Description
FET-D 	**S**	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	Vd	V	Body diode forward voltage drop
	IC		Initial condition: Off/R/Plus/Minus/Diode

FET transistor with body diode. The model is the same as **FET** model with one addition: a body diode connected between drain and source. In each mode of operation, the diode may be open or close, depending on external conditions.

When calculating DC operating point, transistor is set to an initial state specified by **IC** as follows:

Off . . . : I = 0 (open)
R . . . . : V = I * **Rdson** (resistor)
Plus . . : V*ds* > 0, I = (V*gs* - **Vth**) * **S** (“positive” current source)
Minus. : V*ds* < 0, I = (V*gs* - **Vth**) * **S** (“negative” current source)
Diode. : The body diode is conducting.

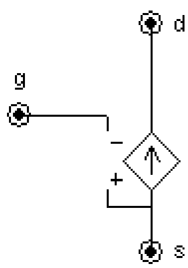
T – P-FET

Symbol	Models	Signals
	Linear Switch FET FET-D SubCir	

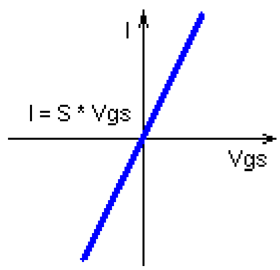
Model	Parameter	Units	Description
Linear	**S**	A/V	Slope
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: drain current.

Linear FET transistor. Voltage controlled current source with specified bandwidth. **S** is open loop slope. Frequency response consists of one pole, **f1** is unit gain frequency. **S** and **f1** can be set to infinity (**inf**).

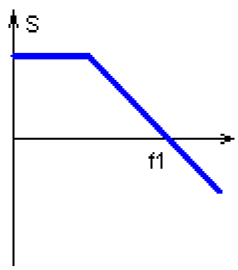
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, drain current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Equivalent schematic



Static characteristic



AC response

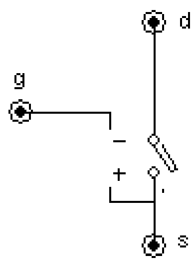
This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

Model	Parameter	Units	Description
Switch	**Vth**	V	Threshold.
	IC		Initial condition of the switch: Off/On.

FET switch. Voltage controlled switch. Switch is closed if gate-source voltage is less than threshold **Vth**.

When calculating DC operating point switch is set to the state specified in **IC**.

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.



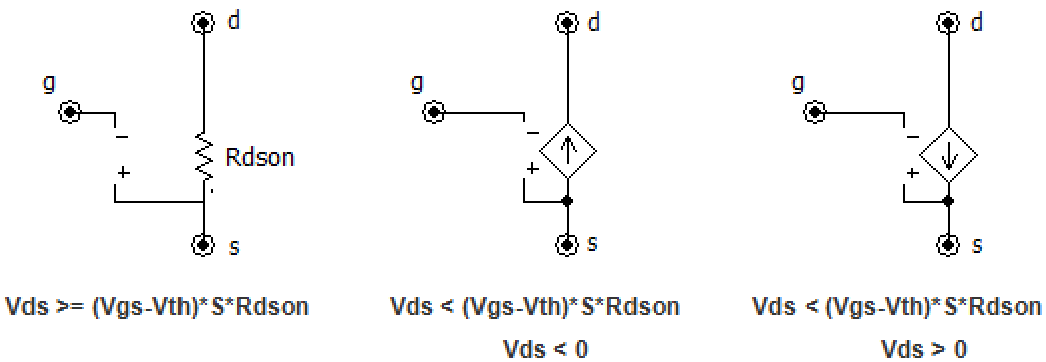
Equivalent schematic

Model	Parameter	Units	Description
FET	**S**	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	IC		Initial condition: Off/R/Plus/Minus

FET transistor. The model has 3 modes of operation.

1. V*gs* >= **Vth** . . . .: I = 0 (open)
2. V*gs* < **Vth**, V*ds* >= (V*gs* - **Vth**) * **S** * **Rdson** . . : V = I * **Rdson** (resistor)
3. V*gs* < **Vth**, V*ds* < (V*gs* - **Vth**) * **S** * **Rdson** . . : I = (V*gs* - **Vth**) * **S** (current source)

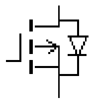
FET works similar for positive and negative drain-source voltage, current direction changes accordingly.
Equivalent schematics (V*gs* < **Vth**):



When calculating DC operating point, transistor is set to an initial state specified by **IC** parameter as follows:

Off . . . : I = 0 (open)
R . . . . : V = I * **Rdson** (resistor)
Plus . . : V*ds* < 0, I = (V*gs* - **Vth**) * **S** ("positive" current source)
Minus. : V*ds* > 0, I = (V*gs* - **Vth**) * **S** ("negative" current source)

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

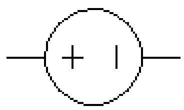
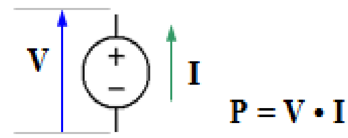
Model	Parameter	Units	Description
FET-D 	**S**	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	Vd	V	Body diode forward voltage drop
	IC		Initial condition: Off/R/Plus/Minus/Diode

FET transistor with body diode. The model is the same as **FET** model with one addition: a body diode connected between drain and source. In each mode of operation, the diode may be open or close, depending on external conditions.

When calculating DC operating point, transistor is set to an initial state specified by **IC** as follows:

Off . . . : I = 0 (open)
R . . . . : V = I * **Rdson** (resistor)
Plus . . : $V_{ds} < 0$, I = (V_{gs} - **Vth**) * **S** ("positive" current source)
Minus. : $V_{ds} > 0$, I = (V_{gs} - **Vth**) * **S** ("negative" current source)
Diode. : The body diode is conducting.

V – Voltage source

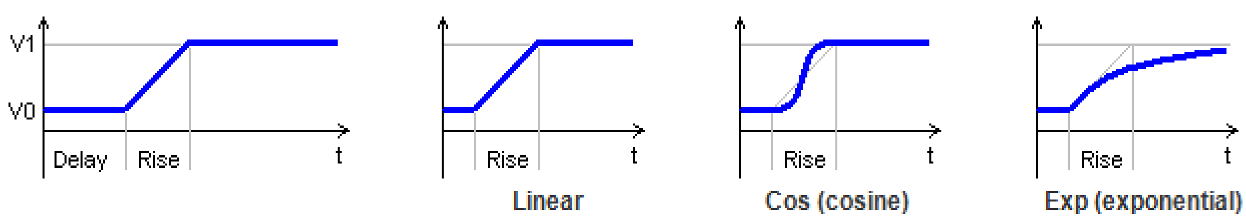
Symbol	Models		Signals
	V Step Single Pulse Clock Sin	Sweep Function List File Trace SubCir	

Model	Parameter	Units	Description
V	**V**	V	Voltage.

Constant voltage = **V**.

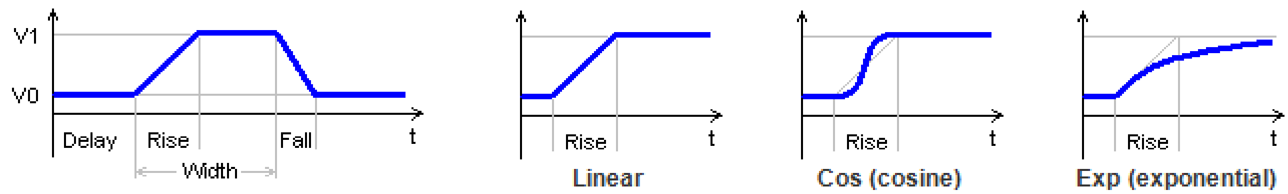
Model	Parameter	Units	Description
Step	**V1**	V	Step On voltage.
	V0	V	Step Off voltage.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

Step starts after **Delay** time. If **Rise** is non-zero, 3 **Slope** types are available.



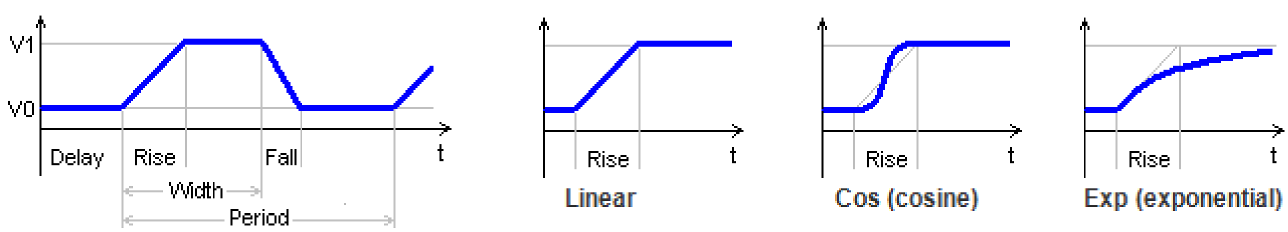
Model	Parameter	Units	Description
Single	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

Single pulse starts after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



Model	Parameter	Units	Description
Pulse	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

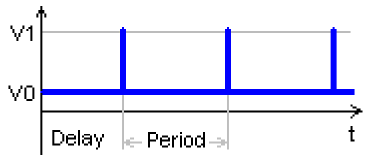
Periodic pulse source. Pulses start after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



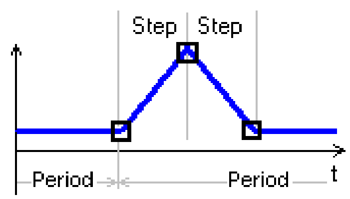
Model	Parameter	Units	Description
Clock	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

Periodic pulses with width of one simulation step. Pulses start after **Delay** time.

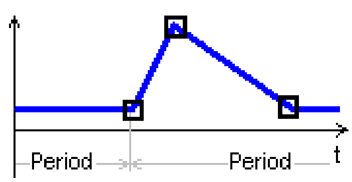
Clock model is recommended to produce a constant frequency clock signal for C-code, DLL, logical components, etc. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:

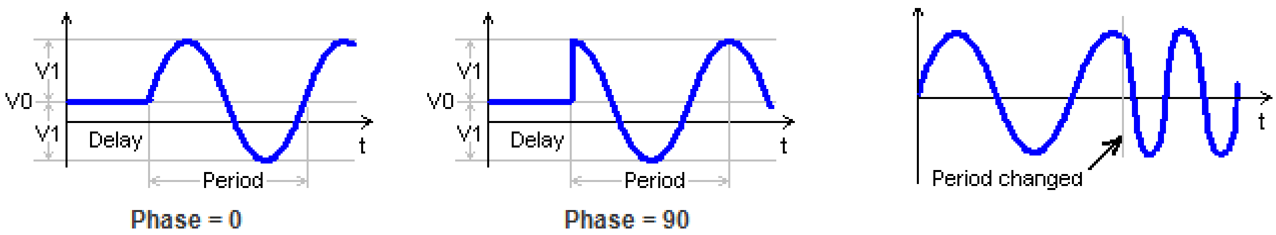


Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:



Model	Parameter	Units	Description
Sin	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

Sinusoidal signal starts after **Delay** time. **Phase** is sine phase in degrees at the moment when signal starts. If transient is paused, sine period changed, and then transient is continued, the phase of the signal remains continuous, providing smooth sine signal of variable frequency. If **Decay** is not zero, the sine signal is exponentially dumped with time constant = 1/**Decay**.



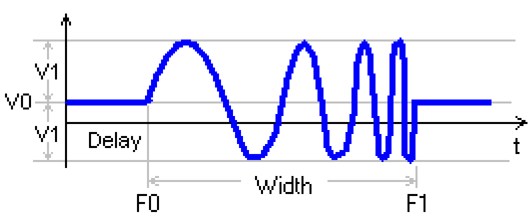
Model	Parameter	Units	Description
Sweep	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

Sinusoidal signal with variable frequency starts after **Delay** time. Signal frequency changes during **Width** interval from **F0** to **F1** linearly or exponentially, depending on specified **Type.**

If **F0** = **F1**, then one period of frequency 1/**Width** will be generated.

If lowest frequency is set to zero and **Type** = Exp, then lowest frequency 0.01/**Width** will be used.

If needed, the highest frequency will be increased to provide integer number of signal periods, so that signal phase at the beginning and at the end of **Width** interval is exactly zero.



Model	Parameter	Units	Description
Function	**f**	V	Function

Arbitrary function. **f** defines voltage as a function of the following variables:

- ***t*** - current time
- **V(*name*)** - voltage on the component ***name***
- **I(*name*)** - current through the component ***name***
- **P(*name*)** – power on the component ***name***
- **S(*name*)** – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, voltage is zero.

Example:

```
f = sin(t) * (1+cos(t*.01))
f = V(R1) * I(R1)
```

Please note that **V**, **I**, **P**, and **S** variables are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined by **List** parameter in the **csv** (comma-separated values) format, as follows:

```
t0,V0,t1,V1,...,tn,Vn
```

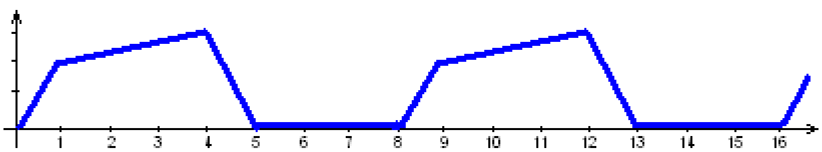
where all t and V can be numerical values or expressions.
If t<t0, signal is V0.
If t0<t<t1, signal value is linearly interpolated between V0 and V1, etc.
If t>tn, and **Cycle** parameter is set to **No**, the signal value is Vn. Otherwise the signal defined in t0…tn interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
List = 0,0,1,2,4,3,5,0,8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following voltage will be generated:



See *Working with List source* chapter for more details.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Signal is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored >
t0,V0
t1,V1
.....
tn,Vn
```

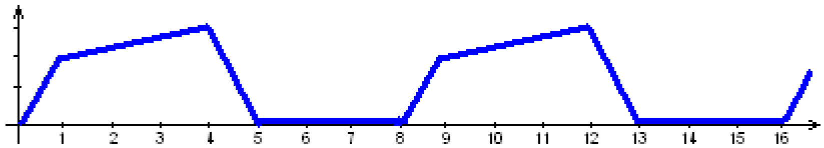
where all t and V can be numerical values or expressions.
If t<t0, signal is V0.
If t0<t<t1, signal value is linearly interpolated between V0 and V1, etc.
If t>tn, and **Cycle** parameter is set to **No**, the signal value is Vn. otherwise the signal defined in t0…tn interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example. File content:

```
0,0
1,2
4,3
5,0
8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following voltage will be generated:



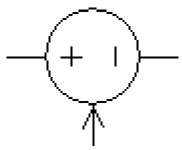
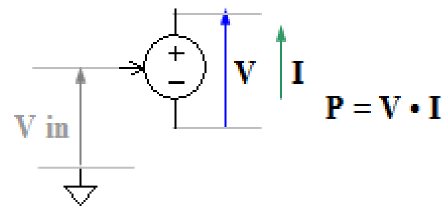
Model	Parameter	Units	Description
Trace	**Trace**		Trace name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Signal is defined by an existing trace. **Trace** parameter is a name of a transient trace. Only traces loaded from data file, imported from text or binary file, duplicated, or pasted from the clipboard can be used.

Signal start is delayed by **Delay** time.

If **Cycle** parameter is set to **Yes**, the signal is repeated continuously.

V – Logic controlled voltage source

Symbol	Models		Signals
	V One-shot Step Single Pulse Clock Sin	Sweep Function List File Trace SubCir	

Model	Parameter	Units	Description
V	**V**	V	Voltage.

Constant voltage = **V**.

Model	Parameter	Units	Description
One-shot	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.

One-shot pulse generator. When increasing input voltage V*in* crosses logical threshold, voltage pulse of **Width** duration is generated. **V0** is pulse Off level, **V1** is pulse On level.

If increasing V*in* crosses logical threshold value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
Step	**V1**	V	Step On voltage.
	V0	V	Step Off voltage.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Step** model of **Voltage source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Single	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Single** model of **Voltage source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Pulse	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Pulse** model of **Voltage source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Clock	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Clock** model of **Voltage source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Sin	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

When control signal V*in* is below logical threshold, output voltage is **V0**. When increasing control signal V*in* crosses logical threshold, a signal similar to **Sin** model of **Voltage source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to **V0** immediately.

Model	Parameter	Units	Description
Sweep	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before first signal starts.

When control signal V*in* is below logical threshold, output voltage is **V0**. When increasing control signal V*in* crosses logical threshold, a signal similar to **Sweep** model of **Voltage source** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to **V0** immediately.

Model	Parameter	Units	Description
Function	**f**	A	Function

When control signal V*in* is below logical threshold, output is zero. When increasing control signal V*in* crosses logical threshold, a signal similar to **Function** model of **Voltage source** component is generated. If the function is using current time variable ***t***, this moment will be considered as ***t***=0. When decreasing control signal V*in* drops below logical threshold, output goes to zero immediately

Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is equal to **V0** value of **List** signal. When increasing control signal V*in* crosses logical threshold, a signal similar to **List** model of **Voltage source** component is generated. This moment is also considered as ***t***=0 for the **List** signal. When decreasing control signal V*in* drops below logical threshold, output goes to **V0** immediately.

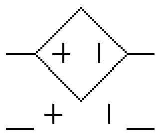
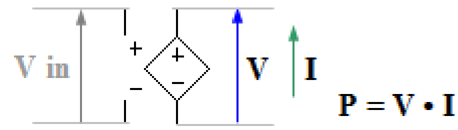
Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is equal to **V0** value specified in the **File**. When increasing control signal V*in* crosses logical threshold, a signal similar to **File** model of **Voltage source** component is generated. This moment is also considered as ***t***=0 for the **File** signal. When decreasing control signal V*in* drops below logical threshold, output goes to **V0** immediately.

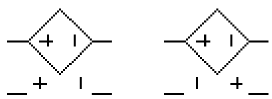
Model	Parameter	Units	Description
Trace	**Trace**		Trace name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is zero. When increasing control signal V*in* crosses logical threshold, a signal similar to **Trace** model of **Voltage source** component is generated. This moment is also considered as ***t***=0 for the **Trace** signal. When decreasing control signal V*in* drops below logical threshold, output goes to zero immediately.

V – Voltage controlled voltage source

Symbol	Models		Signals
	Linear V Function PWL	VCO One-shot PWM SubCir	

Views



Model	Parameter	Units	Description
Linear	**K**	V/V	Gain

Linear voltage controlled voltage source: V = **K** * V*in*.

Model	Parameter	Units	Description
V	**V**	V	Voltage.

Constant voltage = **V**.

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable ***x*** (input voltage V*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(Vin)

Piecewise linear voltage controlled voltage source. **pwl** string defines gain as a function of input voltage K(V*in*). The transfer function of the source V(V*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(V*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
VCO	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline or Off level.
	dFdV	Hz/V	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Voltage controlled oscillator. Output voltage is a signal with frequency equal to:

f(Hz) = **dFdV** * Vin.

For **Sine** signal, **V0** is baseline, and **V1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **V0** is Off level, **V1** is On level. **Phase** is additional phase of the signal, in degrees.

Model	Parameter	Units	Description
One-shot	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Threshold	V	Voltage threshold.

One-shot pulse generator. When increasing input voltage V*in* crosses **Threshold** value, voltage pulse of **Width** duration is generated. **V0** is pulse Off level, **V1** is pulse On level.

If increasing V*in* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
PWM	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	F	Hz	Frequency.
	Vmax	V	Input voltage corresponding to 100% duty.
	Phase	deg	Phase.

Voltage controlled Pulse-Width Modulator. Output voltage is a pulse signal of frequency **F** shifted by **Phase**. Input voltage V*in* is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:

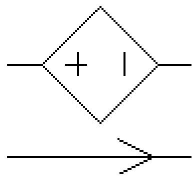
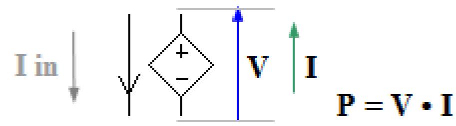
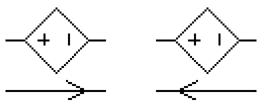
width = 1/**F** * (V*in* / **Vmax**)

or

duty = 100% * (V*in* / **Vmax**);

If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency **F**, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always **F**.

V – Current controlled voltage source

Symbol	Models		Signals
	Linear V Function PWL	CCO One-shot PWM SubCir	
Views: 			

Model	Parameter	Units	Description
Linear	**K**	V/V	Gain

Linear current controlled voltage source. V = **K** * I*in*.

Model	Parameter	Units	Description
V	**V**	V	Voltage.

Constant voltage = **V**.

Model	Parameter	Units	Description
Function	**f**	V	Output as function of the input.
	IC	V	Initial condition: output current.

Arbitrary function f defines output current as a function of the following variables:

x – input current I*in*
t - current time
V(*name*) - voltage on the component ***name***
I(*name*) - current through the component ***name***
P(*name*) – power on the component ***name***
S(*name*) – state of the component ***name***

where ***name*** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

f = x*x
f = x * sin(t)
f = P(r1) + P(r2)

When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable ***x*** (input current I*in*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	**pwl**		Comma-separated string, K(I*in*)

Piecewise linear current controlled voltage source. **pwl** string defines gain as a function of input current K(I*in*). The transfer function of the source V(I*in*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(I*in*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
CCO	**V1**	V	Voltage amplitude.
	V0	V	Voltage baseline or Off level.
	dFdI	Hz/A	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Current controlled oscillator. Output voltage is a signal with frequency equal to:

f(Hz) = **dFdI** * Iin.

For **Sine** signal, **V0** is baseline, and **V1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **V0** is Off level, **V1** is On level. **Phase** is additional phase of the signal, in degrees.

Model	Parameter	Units	Description
One-shot	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Threshold	A	Current threshold.

One-shot pulse generator. When increasing input current I*in* crosses **Threshold** value, voltage pulse of **Width** duration is generated. **V0** is pulse Off level, **V1** is pulse On level.

If increasing I*in* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
PWM	**V1**	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	F	Hz	Frequency.
	Imax	A	Input current corresponding to 100% duty.
	Phase	deg	Phase.

Current controlled Pulse-Width Modulator. Output voltage is a pulse signal of frequency **F** shifted by **Phase**. Input current I*in* is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:

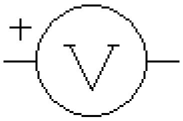
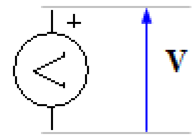
width = 1/**F** * (I*in* / **Imax**)

or

duty = 100% * (I*in* / **Imax**);

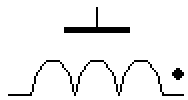
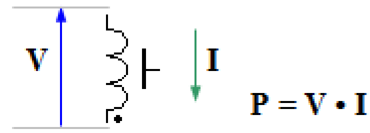
If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency **F**, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always **F**.

V – Voltmeter

Symbol	Models	Signals
	Voltmeter	

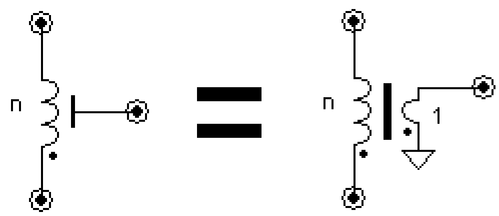
Model	Parameter	Units	Description
Voltmeter	No parameters.		
Open circuit.			

W – Winding

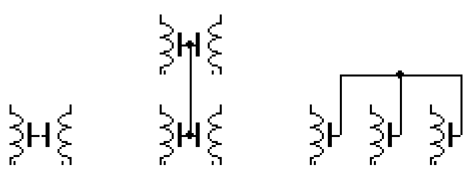
Symbol	Models	Signals
	Winding	

Model	Parameter	Units	Description
Winding	**n**	turns	Number of turns.

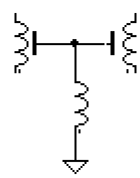
The Winding is actually an ideal transformer, with 1-turn second winding, one end of which is grounded, and another end is shown as a "core" pin of the winding:



To make an ideal transformer, connect cores of two or more windings by wire. Core magnetizing can be modeled by setting linear or non-linear inductor from core to ground:

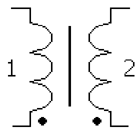
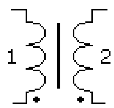
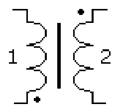


Ideal transformers



Transformer with magnetizing inductor

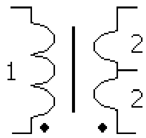
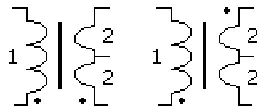
W – Transformer

Symbol	Models	Signals
	Transformer SubCir	
Views	 	

Model	Parameter	Units	Description
Transformer	**n1**	turns	Number of turns in the first winding.
	n2	turns	Number of turns in the second winding.

Ideal transformer with 2 windings. Coupling coefficient = 1.

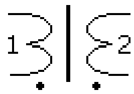
W – Differential transformer

Symbol	Models	Signals
	Transformer SubCir	
Views		

Model	Parameter	Units	Description
Transformer	**n1**	turns	Number of turns in the first winding.
	n2	turns	Number of turns in second and third windings.

Ideal differential transformer with 3 windings. Coupling coefficient = 1. Second and third windings have the same number of turns **n2**, and connected to form a differential transformer.

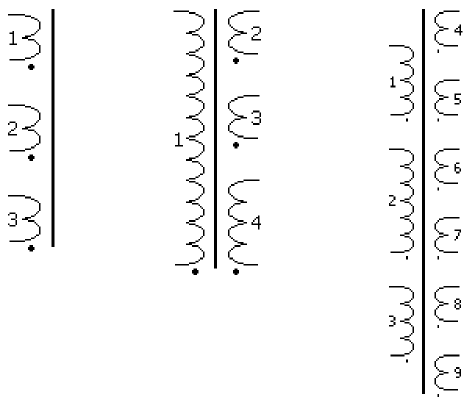
W – Custom transformer

Symbol	Models	Signals
	Transformer SubCir	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:
- height from 2 to 32,
- up to 32 windings (total),
- arbitrary length of a winding.

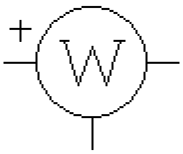
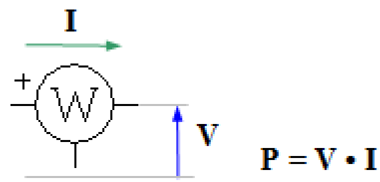
Examples of Custom transformer component:



Model	Parameter	Units	Description
Transformer	**n1**	turns	Number of turns in the first winding.
	…	**…**	**…**
	nN	turns	Number of turns in the Nth winding.

Ideal transformer with N windings. Coupling coefficient = 1.

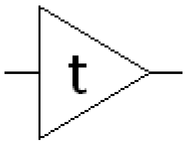
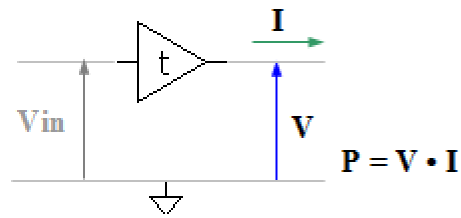
W – Wattmeter

Symbol	Models	Signals
	Wattmeter	

Model	Parameter	Units	Description
Wattmeter	No parameters.		

Short circuit between current ports, open circuit between voltage ports. Can be used to measure power in grounded or non-grounded load.

X – Delay

Symbol	Models	Signals
	Delay SubCir	

Model	Parameter	Units	Description
Delay	**t0**	s	Delay.
	IC	V	Initial condition: output voltage.

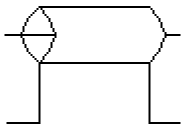
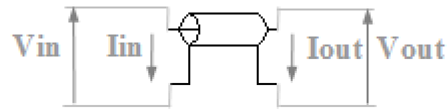
Output voltage is equal to input voltage, delayed by delay time **t0**:

V(t) = V*in*(t – **t0**), where t is current time.

When calculating DC operating point, output is set to specified output voltage **IC**, or to input voltage, if **IC** is blank. Then output voltage is not changing until delay time **t0**.

The model allocates memory for storing delayed data only when needed, and frees it immediately when possible.

X – Transmission line

Symbol	Models	Signals
	Line Lossy	

Model	Parameter	Units	Description
Line	**t0**	s	Delay.
	z0	Ohm	Characteristic impedance.
	VIC	V	Initial condition: voltage.
	IIC	A	Initial condition: current.

Lossless transmission line. The voltage and current in the line are represented as a superposition of forward and reflected waves, with V/I ratio in each wave equal to line characteristic impedance **z0**. V and I values of each wave are calculated based on boundary (input and output) conditions. The line functionality can also be described by the following equations:

V*in*(t) = **z0** * (I*in*(t) – I*out*(t – **t0**))
V*out*(t) = **z0** * (I*out*(t) – I*in*(t – **t0**))

where t is current time.

Input and output are galvanically isolated: no current is flowing between input and output, and any voltage difference between input and output may exist.

When calculating DC operating point, initial forward and reflected voltage and current are calculated based on the following conditions:

if **VIC** and **IIC** are blank . . . . : V*in* = V*out*, I*in* = -I*out*.
if **VIC** is specified and **IIC** is blank . . : V*in* = V*out* = **VIC**.
if **VIC** is blank and **IIC** is specified . . : I*in* = **IIC**, I*out* = - **IIC**.
if **VIC** and **IIC** are specified . . . . : V*in* = V*out* = **VIC**, I*in* = **IIC**, I*out* = - **IIC**.

The model allocates memory for storing forward and reflected wave data only when needed, and frees it immediately when possible.

If real line characteristics are given in line capacitance and inductance per length, the following equations can be used to derive **t0** and **z0** parameters:

t0 = sqrt(L * C) * D
z0 = sqrt(L / C)

where:

C – line capacitance per length, F/m
L – line inductance per length, H/m
D – line length, m

Model	Parameter	Units	Description
Lossy	**t0**	s	Delay.
	z0	Ohm	Characteristic impedance.
	R	Ohm/ns	Series resistance per ns.
	fr	MHz	Skin losses cutoff (3 dB) frequency.
	G	1/Ohm/ns	Shunt conductance per ns.
	fG	MHz	Dielectric losses cutoff (3 dB) frequency.
	VIC	V	Initial condition: voltage.
	IIC	A	Initial condition: current.

Lossy transmission line. Lossy line modeling is similar to lossless transmission line, with addition of losses due to series resistance, skin effect, shunt conductance, and dielectric losses.

Constant series resistance is defined by **r** parameter. Skin losses are modeled by a number of RL chains, providing series impedance increase as a square root of frequency. The number of chains is automatically optimized based on calculation step value; however, the maximum impedance increase due to skin effect is limited to 40 dB (100 times). **fr** parameter defines a frequency where effective series impedance is approximately 3 dB higher than **r**. Skin losses are calculated only if **r** > 0, and **fr** is not infinite.

Constant shunt conductance is defined by **G** parameter. Dielectric losses are modeled by a shunt capacitance, providing shunt admittance increase proportional to frequency. **fG** parameter defines a frequency where effective shunt admittance is approximately 3 dB higher than **G**. Dielectric losses are calculated only if **G** > 0, and **fG** is not infinite.

Input and output are galvanically isolated: no current is flowing between input and output, and any voltage difference between input and output may exist.

When calculating DC operating point initial forward and reflected voltage and current are calculated based on the following conditions:

if **VIC** and **IIC** are blank . . . . : V*in* = V*out*, I*in* = -I*out*.
if **VIC** is specified and **IIC** is blank . . : V*in* = V*out* = **VIC**.
if **VIC** is blank and **IIC** is specified . . : I*in* = **IIC**, I*out* = **- IIC**.
if **VIC** and **IIC** are specified . . . . : V*in* = V*out* = **VIC**, I*in* = **IIC**, I*out* = **- IIC**.

The model allocates all the required memory immediately at transient start. The amount of memory is proportional to line delay and inverse proportional to calculation step.

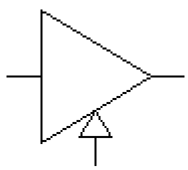
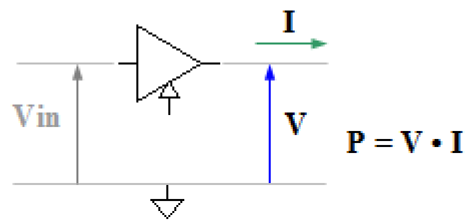
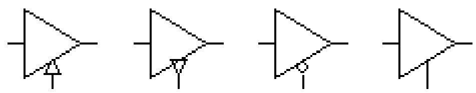
If real line characteristics are given in line capacitance and inductance per length, the following equations can be used to derive **t0** and **z0** parameters:

t0 = sqrt(L * C) * D
z0 = sqrt(L / C)

where:

C – line capacitance per length, F/m
L – line inductance per length, H/m
D – line length, m

X – Sample/hold

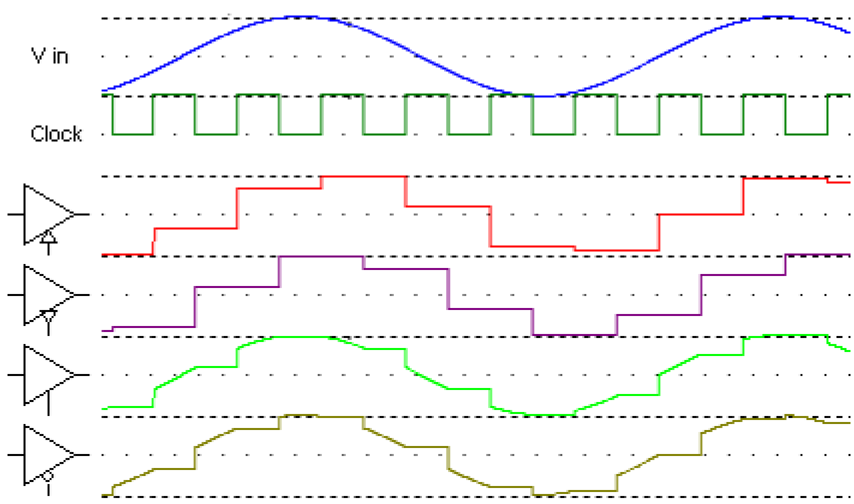
Symbol	Models	Signals
	SH SubCir	
Views		

Model	Parameter	Units	Description
SH	IC	V	Initial condition: output voltage.

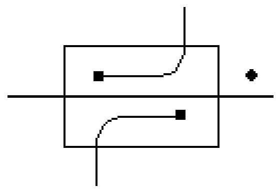
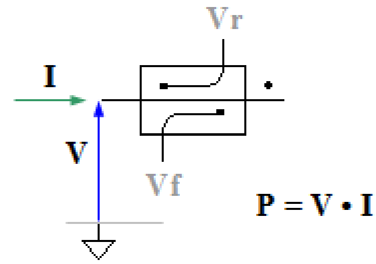
Depending on view, the model is functioning as a sample/hold, or as a track/hold. In sample/hold mode, input voltage is sampled at rising (or falling) edge of a logical clock signal. In track/hold mode, output voltage tracks input voltage while clock signal is above (or below, for inverted control pin) the logical threshold, and holds it while clock signal is below (or above, for inverted control pin) the logical threshold.

When calculating DC operating point output is set to specified output voltage **IC**.

A waveforms example for different modes:



X – Directional coupler

Symbol	Models	Signals
	Coupler	

Model	Parameter	Units	Description
Coupler	**z0**	Ohm	Characteristic impedance
	CF	dB	Coupling factor

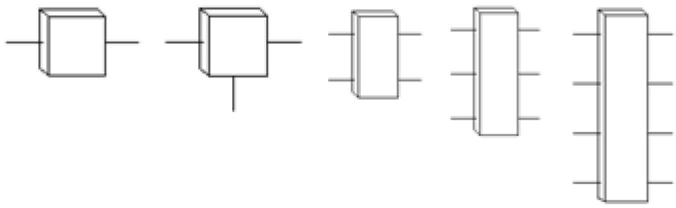
Directional coupler is a short circuit (no insertion loss) with two output ports: forward Vf, and reflected Vr. Output ports are voltage sources with zero output impedance and coupling factor **CF**. The output voltages are calculated as follows:

Vf = K * (V + I * **z0**) / 2
Vr = K * (V – I * **z0**) / 2

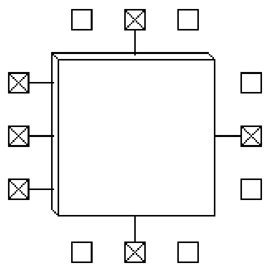
where $K = 10^{-CF/20}$.

All voltages are referenced to ground.

X – Block-2…Block-8

Symbol	Models
	SubCir
Blocks with predefined size and number of pins, to be used as subcircuits (**SubCir** model).	

X – Custom block

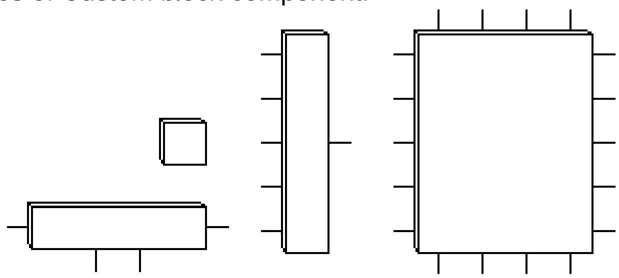
Symbol	Models	Signals
	SubCir	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

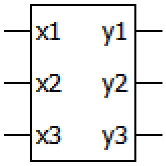
This component may have:

- arbitrary size up to 32(width) X 32(height),
- up to 32 pins on each side

Examples of Custom block component:



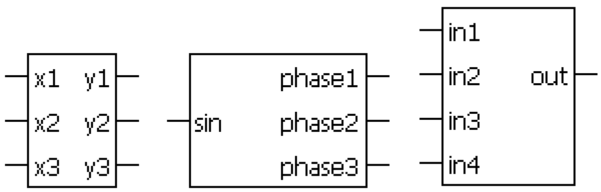
X – NL5 circuit

Symbol	Models	Signals
	SubCir	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:
- arbitrary size up to 32(width) X 256(height),
- up to 256 inputs on the left side,
- up to 256 outputs on the right side,

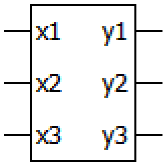
Examples of NL5 circuit component:



Model	Parameter	Units	Description
SubCir	**File**		File name of subcircuit schematic.
	Cmd		Subcircuit start-up command string
	IC		Subcircuit Initial conditions string

This model is similar to a standard **SubCir** model, with one difference: instead of parameters, subcircuit names are entered in the **Edit Component** dialog as a component pin names. See *Working with Subcircuits* chapter for details on subcircuit operation.

X – C-code

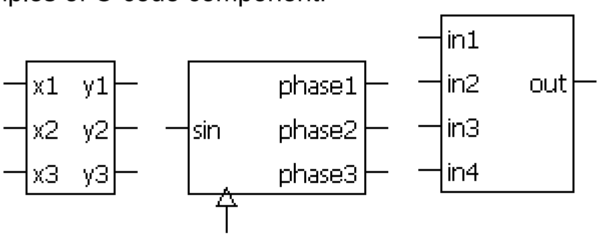
Symbol	Models	Signals
	C File	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 256(height),
- up to 256 inputs on the left side,
- up to 256 outputs on the right side,
- one or no clock pins on the bottom side.
- custom or default input and output names.

Examples of C-code component:



Model	Parameter	Units	Description
C	**Code**		C-code.
	IC		Initial conditions.

C-code block. The model contains code written in simplified C language.

Code contains global variables declaration, initialization code, and main code.

IC may contain assignments of initial values to global variables. If not empty, **IC** will be executed after initialization code.

See *Working with C-code* chapter for details of the model functionality and instructions on creating the code.

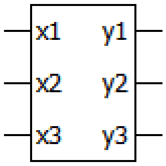
Model	Parameter	Units	Description
File	**File**		C-code file name
	IC		Initial conditions.

The model executes C-code from the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

IC may contain assignments of initial values to global variables. If not empty, **IC** will be executed after initialization code.

See *Working with C-code* chapter for details of the model functionality and instructions on creating the code.

X – DLL

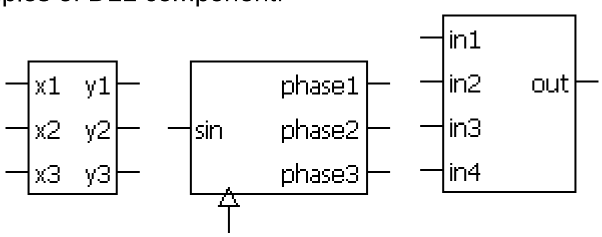
Symbol	Models	Signals
	DLL	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 256(height),
- up to 256 inputs on the left side,
- up to 256 outputs on the right side,
- one or no clock pins on the bottom side.
- custom or default input and output names.

Examples of DLL component:



Model	Parameter	Units	Description
DLL	**DLL**		DLL file name
	Init		Initialization function name.
	Main		Main function name.
	Pause		Pause function name.
	Exit		Exit function name.
	IC		Initial conditions.

DLL block. Component's code is written in C, compiled, and placed in the **64-bit** DLL file. DLL functions will be called by NL5 during transient simulation.

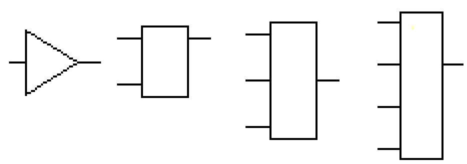
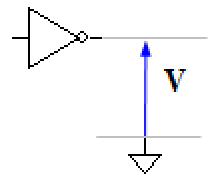
DLL parameter is a DLL file name. If **DLL** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Init - initialization function. Called once at the beginning of transient simulation at t=0. Leave it blank if not used.
Main - main function. Called on every simulation step, or in rising edge of the clock, if clock pin exists.
Pause - called when transient simulation is paused. Leave it blank if not used.
Exit - called when DLL is being closed, and DLL component destroyed. Leave it blank if not used.

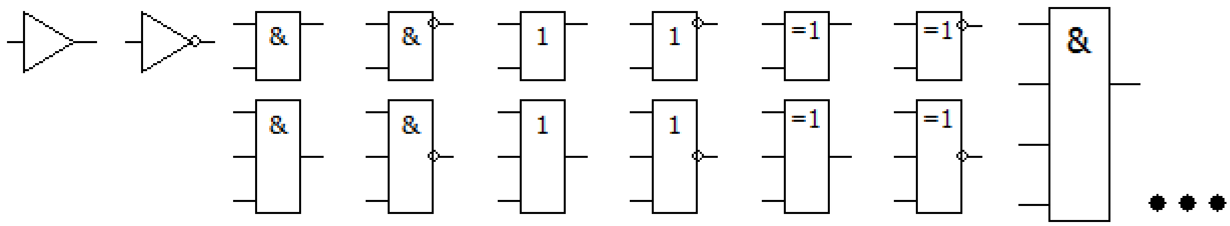
IC may contain assignments of initial values to outputs and component variables. If not empty, **IC** will be executed after initialization code.

See *Working with DLL* chapter for details of the model functionality and instructions on creating code and DLL.

Y – Gates

Symbol	Models	Signals
	Logic Delay Delay2	

Views



For all gates, select **view** for logical function (AND, OR, XOR), and output pin inversion.

XOR (**=1**) function is **odd function** (modulo-2 sum): the output is high if odd number of inputs are high.

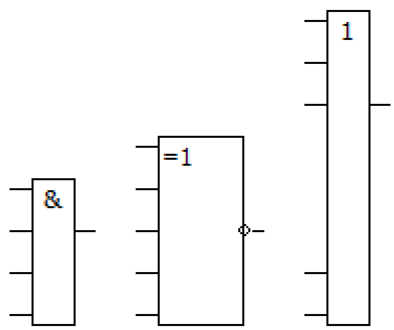
When calculating DC operating point, output is set to specified **IC** level.

Custom gate component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 32(height),
- up to 32 inputs on the left side,
- one output on the right side.

Examples of Custom gate component:

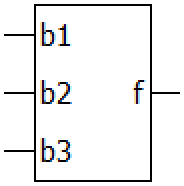
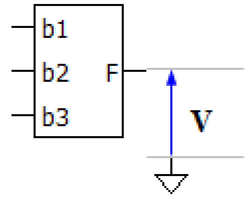


Model	Parameter	Units	Description
Logic	**IC**		Initial condition: Low/High.
Output signal is delayed by one calculation step.			

Model	Parameter	Units	Description
Delay	**Delay**	s	Output delay.
	IC		Initial condition: Low/High.
Output signal is delayed by specified **Delay** time. Short signals (shorter than **Delay**) may not pass through and will not affect output.			

Model	Parameter	Units	Description
Delay2	**Up**	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	IC		Initial condition: Low/High.
Output signal is delayed by **Up** time for rising edge of output signal, and **Down** time for falling edge of output signal. Short signals (comparable or shorter than **Up** or **Down**) may not pass through and will not affect output.			

Y – Logical function

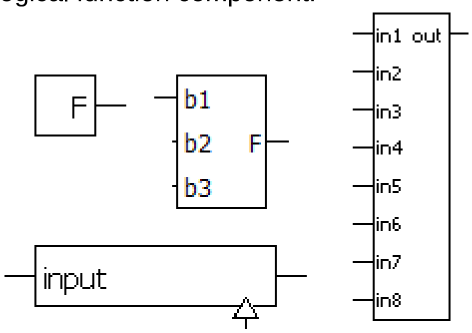
Symbol	Models	Signals
	Function	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 8(height),
- up to 8 inputs on the left side,
- one output on the right side,
- one or no clock pins on the bottom side.
- custom input and output names.

Examples of Logical function component:



Model	Parameter	Units	Description
Function	**f**	V	Output as function of the logical inputs.
	IC	V	Initial condition: Low/High.

Arbitrary logical function **f** defines output logical state as a function of the following variables:

pin_name – logical value of the input voltage on the input pin **pin_name.**
t **-** current time

First, input voltages are converted to logical values, then the function is calculated. A logical result of the function is converted to the output voltage, which may have only Low or High logical level.

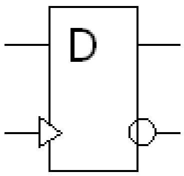
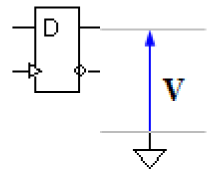
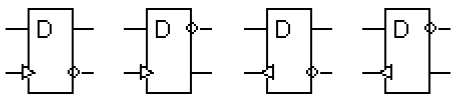
Example:

f = (b1 && b2) || b2
f = selector ? in1 : in2

If ***clock*** pin does not exist, the model operates in **continuous** mode: the function is calculated and applied to the output on every calculation step. If ***clock*** pin exists, the model operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation than **continuous** mode.

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Output voltage is always delayed by one calculation step.

Y – D flip-flop

Symbol	Models	Traces
	Logic Delay Delay2	
Views		
For all models, when calculating DC operating point, output is set to specified **IC** level.		

Model	Parameter	Units	Description
Logic	**IC**		Initial condition: Low/High.

Output signal is delayed by one calculation step.

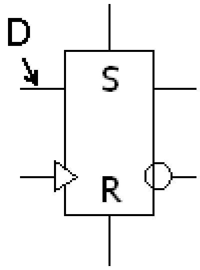
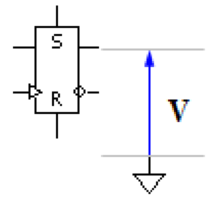
Model	Parameter	Units	Description
Delay	**Delay**	s	Output delay.
	IC		Initial condition: Low/High.

Output signal is delayed by specified **Delay** time.
Short signals (shorter than **Delay**) may not pass through and will not affect output.

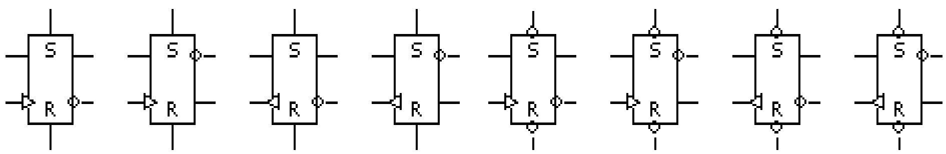
Model	Parameter	Units	Description
Delay2	**Up**	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	IC		Initial condition: Low/High.

Output signal is delayed by **Up** time for rising edge of output signal, and **Down** time for falling edge of output signal. Short signals (comparable or shorter than **Up** or **Down**) may not pass through and will not affect output.

Y – SR trigger

Symbol	Models	Traces
	Logic Delay Delay2	

Views



For all models, For all models, when calculating DC operating point, output is set to specified **IC** level.
Dominance defines trigger output states in case both Set and Reset inputs are active.
Please note that 'D' input is not shown on the symbol due to limited space.

Model	Parameter	Units	Description
Logic	**Dominance**		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.

Output signal is delayed by one calculation step.

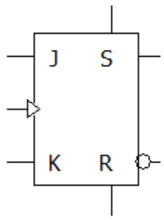
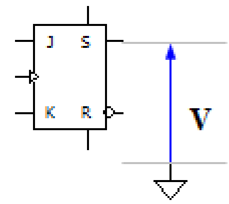
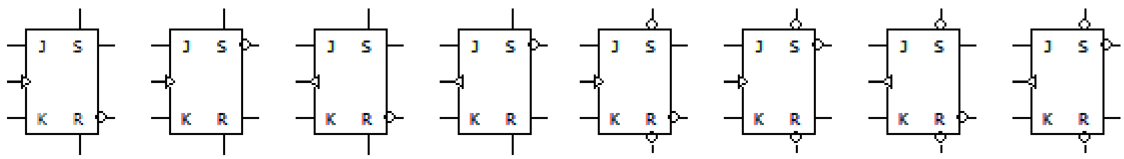
Model	Parameter	Units	Description
Delay	**Delay**	s	Output delay.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.

Output signal is delayed by specified **Delay** time.
Short signals (shorter than **Delay**) may not pass through and will not affect output.

Model	Parameter	Units	Description
Delay2	**Up**	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.

Output signal is delayed by **Up** time for rising edge of output signal, and **Down** time for falling edge of output signal. Short signals (comparable or shorter than **Up** or **Down**) may not pass through and will not affect output.

Y – JK trigger

Symbol	Models	Traces
	Logic Delay Delay2	
Views		

For all models, For all models, when calculating DC operating point, output is set to specified **IC** level. **Dominance** defines trigger output states in case both Set and Reset inputs are active.

Model	Parameter	Units	Description
Logic	**Dominance**		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.

Output signal is delayed by one calculation step.

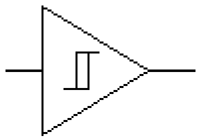
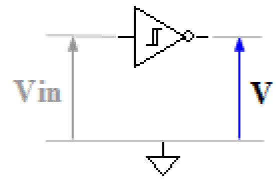
Model	Parameter	Units	Description
Delay	**Delay**	s	Output delay.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.

Output signal is delayed by specified **Delay** time.
Short signals (shorter than **Delay**) may not pass through and will not affect output.

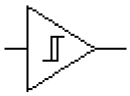
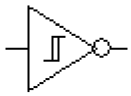
Model	Parameter	Units	Description
Delay2	**Up**	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.

Output signal is delayed by **Up** time for rising edge of output signal, and **Down** time for falling edge of output signal. Short signals (comparable or shorter than **Up** or **Down**) may not pass through and will not affect output.

Y – Schmitt trigger

Symbol	Models	Traces
	Logic Delay Delay2	

Views

For all models, when calculating DC operating point, output is set to specified **IC** level.

Output is set to Low or High level as follows (non-inverted output, **Threshold** is a logical level threshold):

V*in* > **Threshold** + **Hysteresis**/2 . . . : V = High
V*in* < **Threshold** - **Hysteresis**/2 . . . .: V = Low
Otherwise . . . .: V = previous state

Model	Parameter	Units	Description
Logic	**Hysteresis**	V	Hysteresis.
	IC		Initial condition: Low/High.

Output signal is delayed by one calculation step.

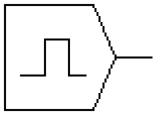
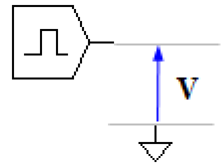
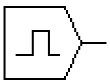
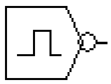
Model	Parameter	Units	Description
Delay	**Hysteresis**	V	Hysteresis.
	Delay	s	Output delay
	IC		Initial condition: Low/High.

Output signal is delayed by specified **Delay** time.
Short signals (shorter than **Delay**) may not pass through and will not affect output.

Model	Parameter	Units	Description
Delay2	**Hysteresis**	V	Hysteresis.
	Up	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	IC		Initial condition: Low/High.

Output signal is delayed by **Up** time for rising edge of output signal, and **Down** time for falling edge of output signal. Short signals (comparable or shorter than **Up** or **Down**) may not pass through and will not affect output.

Y – Logic generator

Symbol	Models		Traces
	Low High Step Single	Pulse Clock List File	
Views	 		

Model	Parameter	Units	Description
Low	No parameters.		

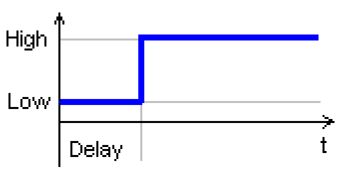
Output is always **Low**, regardless of output inversion state.

Model	Parameter	Units	Description
High	No parameters.		

Output is always **High**, regardless of output inversion state

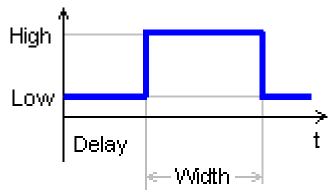
Model	Parameter	Units	Description
Step	**Delay**	s	Delay before active state.

Output goes to active state after **Delay** time. The following signal will be generated for non-inverted output:



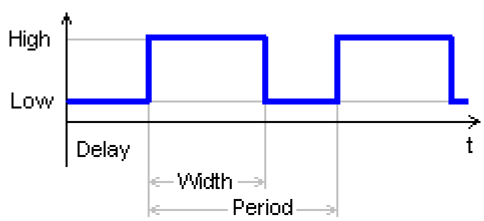
Model	Parameter	Units	Description
Single	**Width**	s	Pulse width.
	Delay	s	Delay before first pulse starts.

The pulse starts after **Delay** time. The following signal will be generated for non-inverted output:



Model	Parameter	Units	Description
Pulse	**Period**	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.

Pulses start after **Delay** time. The following signal will be generated for non-inverted output:

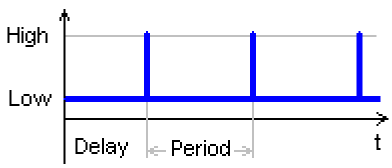


Model	Parameter	Units	Description
Clock	**Period**	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

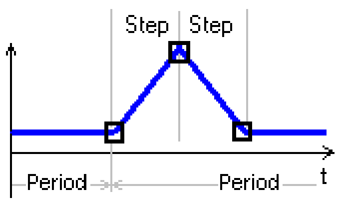
Pulses start after **Delay** time. Output goes to active state for one simulation step only.

Clock model is recommended to produce a constant frequency clock signal for C-code, DLL, logical components, etc. Unlike **Pulse** model, it won’t force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.

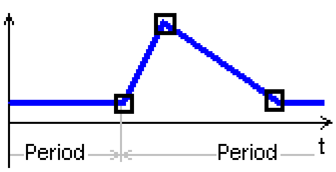
The following signal will be generated for non-inverted output:



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:



Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:



Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Output sequence is defined in the **List** parameter in the **csv** (comma-separated values) format, as follows:

t0,s0,t1,s1,...,tn,sn

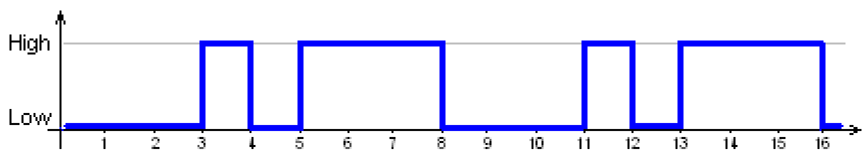
s0…sn defines output logical level: positive number corresponds to High, zero or negative number to Low.
If t<t0, output level is s0.
At t0 output level is s0, at t1 output level is s1, and so on.
If t>tn, and **Cycle** parameter is set to **No**, output level remains at sn. Otherwise the sequence is repeated continuously.

Sequence start is delayed by **Delay** time.

Example:

```
List = 0,0,3,1,4,0,5,1,8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following logical output will be generated:



See *Working with List model* chapter for more details.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Output sequence defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Logical output sequence is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored>
t0,s0
t1,s1
.....
tn,sn
```

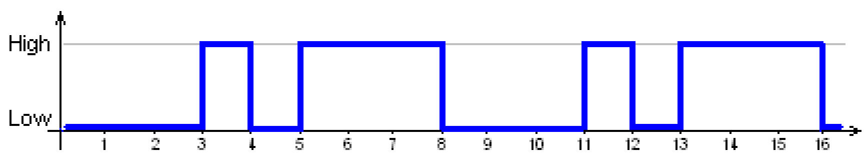
s0…sn defines output logical level: positive number corresponds to High, zero or negative number to Low.
If t<t0, output level is s0.
At t0 output level is s0, at t1 output level is s1, and so on.
If t>tn, and **Cycle** parameter is set to **No**, output level remains at sn. Otherwise the sequence is repeated continuously.

Sequence start is delayed by **Delay** time.

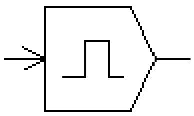
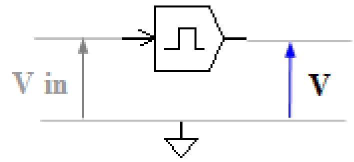
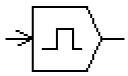
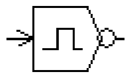
Example:

```
0,0
3,1
4,0
5,1
8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following sequence will be generated:



Y – Logic controlled logic generator

Symbol	Models		Signals
	Gate Low High One-shot Step	Single Pulse Clock List File	
Views	 		

Model	Parameter	Units	Description
Gate	**IC**		Initial condition: Low/High.

Component operates similar to **Logic** model of **Gate** component.

When calculating DC operating point, output is set to the state defined in **IC**.

Model	Parameter	Units	Description
Low	No parameters.		

Output is always **Low**, regardless of output inversion state.

Model	Parameter	Units	Description
High	No parameters.		

Output is always **High**, regardless of output inversion state

Model	Parameter	Units	Description
One-shot	**Width**	s	Pulse width.

One-shot pulse generator. When increasing input voltage V*in* crosses logical threshold, logical pulse of **Width** duration is generated.

If increasing V*in* crosses logical threshold value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
Step	**Delay**	s	Delay before active state.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Step** model of **Logic generator** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Single	**Width**	s	Pulse width.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Single** model of **Logic generator** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Pulse	**Period**	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Pulse** model of **Logic generator** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Clock	**Period**	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

When control signal V*in* is below logical threshold, output is in Off state. When increasing control signal V*in* crosses logical threshold, a signal similar to **Clock** model of **Logic generator** component is generated. When decreasing control signal V*in* drops below logical threshold, output goes to Off state immediately.

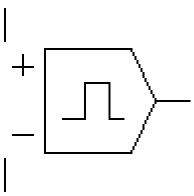
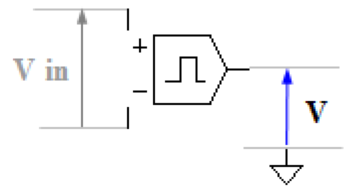
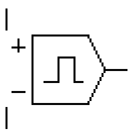
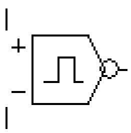
Model	Parameter	Units	Description
List	**List**		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is equal to **s0** value of **List** signal. When increasing control signal V*in* crosses logical threshold, a signal similar to **List** model of **Logic generator** component is generated. This moment is also considered as ***t***=0 for the **List** signal. When decreasing control signal V*in* drops below logical threshold, output goes to **s0** immediately.

Model	Parameter	Units	Description
File	**File**		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V*in* is below logical threshold, output is equal to **s0** value specified in the **File**. When increasing control signal V*in* crosses logical threshold, a signal similar to **File** model of **Logic generator** component is generated. This moment is also considered as ***t***=0 for the **File** signal. When decreasing control signal V*in* drops below logical threshold, output goes to **s0** immediately.

Y – Voltage controlled logic generator

Symbol	Models	Signals
	Gate Low High One-shot	
Views	 	

Model	Parameter	Units	Description
Gate	**Threshold**	V	Voltage threshold.
	Hysteresis	V	Hysteresis.
	IC		Initial condition: Low/High.

Gate with hysteresis. Output is set to Low or High level as follows (non-inverted output):

V*in* > **Threshold** + **Hysteresis**/2 . . . :   V = High
V*in* < **Threshold** - **Hysteresis**/2 . . . .:   V = Low
Otherwise . . . .:   V = previous state

When calculating DC operating point, output is set to the state defined in **IC**.

Model	Parameter	Units	Description
Low	No parameters.		

Output is always **Low**, regardless of output inversion state.

Model	Parameter	Units	Description
High	No parameters.		

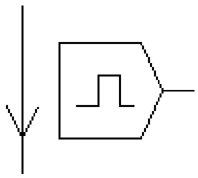
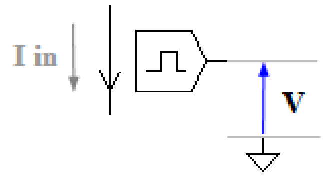
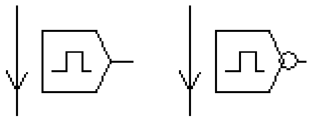
Output is always **High**, regardless of output inversion state

Model	Parameter	Units	Description
One-shot	**Threshold**	V	Voltage threshold.
	Width	s	Pulse width.

One-shot pulse generator. When increasing input voltage V*in* crosses **Threshold**, logical pulse of **Width** duration is generated.

If increasing V*in* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Y – Current controlled logic generator

Symbol	Models	Signals
	Gate Low High One-shot	
Views		

Model	Parameter	Units	Description
Gate	**Threshold**	A	Current threshold.
	Hysteresis	A	Hysteresis.
	IC		Initial condition: Low/High.

Gate with hysteresis. Output is set to Low or High level as follows (non-inverted output):

I*in* > **Threshold** + **Hysteresis**/2 . . . : V = High
I*in* < **Threshold** - **Hysteresis**/2 . . . .: V = Low
Otherwise . . . . : V = previous state

When calculating DC operating point, output is set to the state defined in **IC**.

Model	Parameter	Units	Description
Low	No parameters.		

Output is always **Low**, regardless of output inversion state.

Model	Parameter	Units	Description
High	No parameters.		

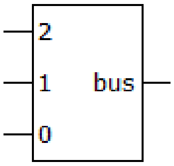
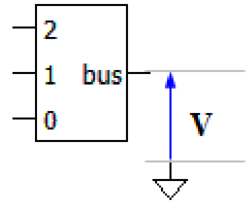
Output is always **High**, regardless of output inversion state

Model	Parameter	Units	Description
One-shot	**Threshold**	A	Current threshold.
	Width	s	Pulse width.

One-shot pulse generator. When increasing input current I*in* crosses **Threshold**, logical pulse of **Width** duration is generated.

If increasing I*in* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Y – Bus

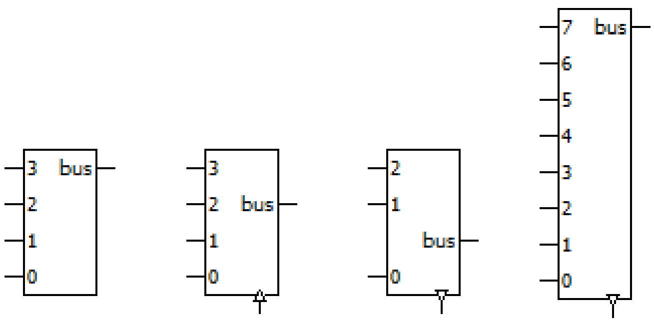
Symbol	Models	Traces
	Bus	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 32(height),
- up to 32 inputs on the left side,
- one output on the right side,
- one or no clock pins on the bottom side.

Examples of **Bus** component:

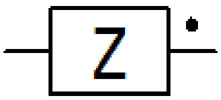
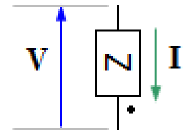


Model	Parameter	Units	Description
Bus	**Format**		Bus format: Signed/Unsigned
	IC	V	Initial condition: output voltage.

Digital-to-bus converter. Converts logical inputs to a bus signal, considering logical inputs as bits of binary number. Input **0** is **LSB**. If **Format** is set to Signed, an input number is considered to be in two's complement format.

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Output voltage is always delayed by one calculation step.

Z – Impedance

Symbol	Models		Signals
	Function Poly1 Poly2 Poly3	Poly4 Poly5 Roots	 $P = V \cdot I$

Model	Parameter	Units	Description
Function	**f**	Ohm	Impedance.

Arbitrary function **f** defines impedance in ***s*** domain. The following variables can represent frequency in the function:

f – current AC frequency, Hz
w – angular AC frequency, ***w = 2πf*** .
s or ***p*** – Laplace parameter, ***s = p = j*2πf***.

Example:

f = 1/(1+s)
f = exp(-R1*C1**s)

Only operators and functions that support complex numbers can be used in this function. If **f** is blank, it is assumed to be zero.

At transient and DC operation point calculation for AC (if enabled), the impedance is equal to **f(0)**.

Model	Parameter	Units	Description
Poly1 **Poly2** **Poly3** **Poly4** **Poly5**	**b0**		Numerator polynomial coefficients 0.
	…		…
	a0		Denominator polynomial coefficients 0.
	…		…
	IC		Initial condition.

Impedance is a ratio of polynomials of Laplace parameter s:

f(s) = (**b0** + **b1***s + **b2***s^2 + …) / (**a0** + **a1***s + **a2***s^2 + …)

These models support transient as well.

Initial condition **IC** consists of internal model values. It should not be manually edited, except clearing it to blank (no **IC**).

At DC operation point calculation, **f(0)** is used.

Model	Parameter	Units	Description
Roots	**K**		Gain.
	Roots		Roots (zeroes and poles).
	IC	V	Initial condition..

Impedance is defines by zeroes and poles:

f(s) = **K** * (s-z1)*(s-z2)… /(s-p1)/(s-p2)…

where **K** is gain, z1…zn are zeroes, p1…pN are poles.

Roots are defined by **Roots** parameter in the **csv** (comma-separated values) format, as follows:

Nz,Rez1,Imz1,…,Np,Rep1,Imp1,…

where:

Nz - number of zeroes
Rezi – real part of zi
Imzi – imaginary part of zi
Np - number of poles,
Repi – real part of pi
Impi – imaginary part of pi

There could be any number of zeroes and poles, however the resulting numerator and denominator polynomials order should not exceed 5. See *Working with Roots model* chapter for details on entering/editing roots.

The model supports transient as well.

Initial condition **IC** consists of internal model values. It should not be manually edited, except clearing it to blank (no **IC**).

At DC operation point calculation, **f(0)** is used.

Operators

Operators are listed in descending precedence order (1 - most, 14 - least).
The table is based on *http://en.cppreference.com/w/cpp/language/operator_precedence*

<table>
<tr><th>Precedence</th><th>Operator</th><th>Description</th></tr>
<tr><td>1</td><td>()
[]
x++
x--
++x
--x</td><td>Function call
Array subscripting
Postfix increment: x=x+1 after use
Postfix decrement: x=x-1 after use
Prefix increment: x=x+1 before use
Prefix decrement: x=x-1 before use</td></tr>
<tr><td>2</td><td>+
-
!
~
(bool)
(int)
(int64)
(float)
(double)
(complex)</td><td>Unary plus
Unary minus
Logical NOT
Bitwise NOT
Type cast to bool
Type cast to int
Type cast to int64
Type cast to float
Type cast to double
Type cast to complex</td></tr>
<tr><td>3</td><td>*
/
%</td><td>Multiplication
Division
Remainder</td></tr>
<tr><td>4</td><td>+
-</td><td>Addition
Subtraction</td></tr>
<tr><td>5</td><td><<
>></td><td>Bitwise left shift
Bitwise right shift</td></tr>
<tr><td>6</td><td><
<=
>
>=</td><td>Relation operator "less than"
Relation operator "less than or equal to"
Relation operator "greater than"
Relation operator "greater than or equal to"</td></tr>
<tr><td>7</td><td>==
!=</td><td>Relation operator "equal to"
Relation operator "not equal to"</td></tr>
<tr><td>8</td><td>&</td><td>Bitwise AND</td></tr>
<tr><td>9</td><td>^</td><td>Bitwise XOR (exclusive OR)</td></tr>
<tr><td>10</td><td>|</td><td>Bitwise OR</td></tr>
<tr><td>11</td><td>&&</td><td>Logical AND</td></tr>
<tr><td>12</td><td>||</td><td>Logical OR</td></tr>
<tr><td>13</td><td>?:</td><td>Ternary conditional operator</td></tr>
<tr><td>14</td><td>=
+=
-=
*=
/=
%=
<<=
>>=
&=
^=
|=</td><td>Assignment
Assignment by sum
Assignment by difference
Assignment by product
Assignment by quotient
Assignment by remainder
Assignment by bitwise left shift
Assignment by bitwise right shift
Assignment by bitwise AND
Assignment by bitwise XOR
Assignment by bitwise OR</td></tr>
</table>

Functions

abs, mag

Prototype	***double*** `abs(`***complex***`)` ***double*** `abs(`***double***`)` ***int64*** `abs(`***int64***`)` ***int*** `abs(`***int***`)`
Description	Absolute value (magnitude). For complex argument: abs = $\sqrt{re^2 + im^2}$. `mag()` can be used instead of `abs()`.

Examples	`abs(3.0+4.0j) = 5.0`
	`abs(-3j) = 3.0`
	`abs(1.0) = 1.0`
	`abs(-10) = 10`

sign

Prototype	***int*** `sign(`***double***`)`
Description	Indicates whether a numeric value is positive, negative, or zero. `sign(x)` returns: • `0 if x=0` • `1 if x>0` • `-1 if x<0`

Examples	`sign(1.234) = 1`
	`sign(0) = 0`
	`sign(-5) = -1`

re, im

Prototype	***double*** `re(`***complex***`)` ***double*** `im(`***complex***`)`
Description	Real and imaginary part of complex number.
Examples	`re(1.2+3.4j) = 1.2` `im(1.2+3.4j) = 3.4`

phase

Prototype	***double*** phase(***complex***)
Description	Phase of complex number. Returns phase in the range –Pi…+Pi.
Examples	phase(1+1j) = 0.785398 (Pi/4)

sqrt

Prototype	***complex*** sqrt(***complex***) ***double*** sqrt(***double***)
Description	Square root. If argument is double, negative argument will cause error.
Examples	sqrt(4.0) = 2 sqrt(-4.0) : math error sqrt(2j) = 1+1j

sqr

Prototype	***double*** sqr(***double***)
Description	"Signed" square root. sqr(x) returns: • $\sqrt{x}$ if x>=0 • $-\sqrt{(-x)}$ if x<0
Examples	sqr(4) = 2 sqr(-4) = -2

sq

Prototype	***complex*** sq(***complex***) ***double*** sq(***double***)
Description	sq(x) calculates x*x : square of the argument.
Examples	sq(2) = 4 sq(1+1j) = 0+2j

lim, limit

Prototype	***double*** `lim(`***double*** `x,` ***double*** `min,` ***double*** `max)`
Description	Limiting function. `lim(x, min, max)` returns: • `x, if min<=x<=max` • `min, if x<min` • `max, if x>max` `limit()` can be used instead of `lim()`.

Examples	`lim(0,-1,2 ) = 0`
	`lim(-2,-1,2) = -1`
	`lim(10,-1,2) = 2`

islow, ishigh

Prototype	***bool*** `islow(`***double***`)` ***bool*** `ishigh(`***double***`)`
Description	Compares argument with logical threshold. `islow(x)` returns `true if x` is less than circuit logical threshold, otherwise `false`. `ishigh(x)` returns `true if x` is greater than circuit logical threshold, otherwise `false`.
Examples	`islow(1.0) = true` `ishigh(1.0) = false`

sum

Prototype	***complex*** `sum(`***complex,...***`)` ***complex*** `sum(`***complex[]***`)` ***double*** `sum(`***double,...***`)` ***double*** `sum(`***double[]***`)`
Description	`sum(x,...)` returns sum of arguments. Number of arguments is not limited. If `x` is an array `x[N]`, `sum(x)` returns sum of all array elements.

Examples	`sum(1.0,2.0,3.0) = 6.0`
	`sum(1.0+1.0j,2.0+2.0j) = 3.0+3.0j`
	`double x[] = { 1.0, 2.0, 3.0, 4.0 };`
	`sum(x) = 10.0;`

mean, average

Prototype	***complex*** `mean(`***complex,...***`)` ***complex*** `mean(`***complex[]***`)` ***double*** `mean(`***double,...***`)` ***double*** `mean(`***double[]***`)`
Description	`mean(x,...)` returns mean (average) value of arguments. Number of arguments is not limited. If `x` is an array `x[N]`, `sum(x)` returns mean (average) value of all array elements. `average()` can be used instead of `mean()`.

Examples	`mean(1.0,2.0,3.0) = 2.0`
	`mean(1.0+1.0j,2.0+2.0j) = 1.5+1.5j`
	`double x[] = { 1.0, 2.0, 3.0, 4.0 };`
	`mean(x) = 2.5;`

min

Prototype	***double*** `min(`***double,...***`)` ***double*** `min(`***double[]***`)` ***int64*** `min(`***int64,...***`)` ***int64*** `min(`***int64[]***`)` ***int*** `min(`***int,...***`)` ***int*** `min(`***int[]***`)` ***bool*** `min(`***bool,...***`)` ***bool*** `min(`***bool[]***`)`
Description	`min(x,...)` returns smaller value of arguments. Number of arguments is not limited. If `x` is an array `x[N]`, `min(x)` returns smaller value of all array elements.

Examples	`min(1.0,2.0,3.0) = 1.0`
	`min(1,2,3) = 1`
	`min(false, true, true) = false`
	
	`double x[] = { -1.0, 2.0, -3.0, 4.0 };`
	`min(x) = -3.0;`

max

Prototype	***double*** `max(`***double,...***`)` ***double*** `max(`***double[]***`)` ***int64*** `max(`***int64,...***`)` ***int64*** `max(`***int64[]***`)` ***int*** `max(`***int,...***`)` ***int*** `max(`***int[]***`)` ***bool*** `max(`***bool,...***`)` ***bool*** `max(`***bool[]***`)`
Description	`max(x,...)` returns larger value of arguments. Number of arguments is not limited. If `x` is an array `x[N]`, `max(x)` returns larger value of all array elements.

Examples	`max(1.0,2.0,3.0) = 3.0`
	`max(1,2,3) = 3`
	`max(false, true, true) = true`
	
	`double x[] = { -1.0, 2.0, -3.0, 4.0 };`
	`max(x) = 4.0;`

exp

Prototype	***complex*** `exp(`***complex***`)` ***double*** `exp(`***double***`)`
Description	`exp(x)` calculates the exponential `e` to the `x`.
Examples	`exp(3.0) = 20.0855` `exp(PI*0.5j) = 0+1j`

pow

Prototype	***complex*** `pow(`***complex*** `x,` ***double*** `y)` ***double*** `pow(`***double*** `x,` ***double*** `y)`
Description	`pow(x,y)` calculates x^y : `x` to the power of `y`. If double argument `x` is negative, math error may occur.

Examples	`pow(10.0,2.0) = 100.0`
	`pow(1j,3) = 0-1j`
	`pow(-4.0,0.5) : math error`
	`pow(-4.0+0j,0.5) = 0+2j`

pwr

Prototype	***double*** `pwr(`***double*** `x,` ***double*** `y)`
Description	"Signed" power function. `pwr(x,y)` returns: • x^y if `x>=0`, • $-(-x)^y$ if `x<0`
Examples	`pwr(10.0,2.0) = 100.0` `pwr(-10.0,2.0) = -100.0`

log(x,y)

Prototype	***complex*** `log(`***complex*** `x,` ***double*** `y)` ***double*** `log(`***double*** `x,` ***double*** `y)`
Description	Calculates logarithm `x` to base `y`.

Examples	`log(128,2) = 7`
	`log(PI,PI) = 1.0`
	`log(-10.0,10.0) : math error`
	`log(-10.0+0j,10.0) = 1+1.36437j`
	`log(1j,10.0) = 0+682.1e-3j`

ln, log

Prototype	***complex*** `ln(`***complex***`)` ***double*** `ln(`***double***`)`
Description	Calculates the natural logarithm. `log()` with one argument can be used instead of `ln()`.

Examples	`ln(100) = 4.60517`
	`ln(-1.0) : math error`
	`ln(-1.0+0j) = 0+3.14159j`

lg, log10

Prototype	***complex*** `lg(`***complex***`)` ***double*** `lg(`***double***`)`
Description	Calculates logarithm to base ten. `log10()` can be used instead of `lg()`.

Examples	`lg(100.0) = 2`
	`lg(-100.0) : math error`
	`lg(-100.0+0j) = 2+1.36437j`

lb, log2

Prototype	***complex*** lb(***complex***) ***double*** lb(***double***)
Description	Calculates logarithm to base two. `log2()` can be used instead of `lb()`.

Examples	`lb(128) = 7`
	`lb(-8.0) : math error`
	`lb(-8.0+0j) = 3+4.53236j`

db

Prototype	***double*** db(***double***) ***double*** db(***double*** x, ***double*** y)
Description	`db(x)` calculates value of `x` in decibel, as: $20*\log_{10}(abs(x))$ `db(x,y)` calculates value of the ratio `x/y` in decibel, as: $20*\log_{10}(abs(x/y))$
Examples	`db(100)=40` `db(0.1,20.0) = -46.0205999133`

par

Prototype	***complex*** par(***complex,...***) ***double*** par(***double,...***)
Description	Parallel connection of real or complex impedances. Number of arguments is not limited.
Examples	`par(1.0,1.0) = 0.5` `par(1.0,2.0,3.0,4.0) = par(par(1.0,2.0),par(3.0,4.0)) = 0.48`

sin, cos, tan, tg

Prototype	***double*** sin(***double***) ***double*** cos(***double***) ***double*** tan(***double***)
Description	Calculates sine, cosine, tangent. `tg()` can be used instead of `tan()`.

Examples	`sin(1.570796327) = 1.0`
	`cos(1.570796327) = 0.0`
	`tan(0.78539816339) = 1.0`

asin, acos, atan

Prototype	***double*** `asin(`***double***`)` ***double*** `acos(`***double***`)` ***double*** `atan(`***double***`)`
Description	Calculates arcsine, arccosine, arctangent. `asin` returns angle in the range `–Pi/2…+Pi/2` `acos` returns angle in the range 0…`Pi`. `atan` returns angle in the range `–Pi/2…+Pi/2`.

Examples	`asin(1.0) = 1.57079632679`
	`acos(1.0) = 0`
	`atan(1.0) = 0.785398163397`

atan2

Prototype	***double*** `atan2(`***double*** `x,` ***double*** `y)`
Description	Calculates arctangent of `x/y`. Returns angle in the range `–Pi…Pi`.
Examples	`atan2(1.0,1.0) = 0.785398163397`

random, rand

Prototype	***double*** `random(`***double***`)`
Description	`random(x)` returns random number with uniform distribution in the range `0…x`. `rand()` can be used instead `of random()`.
Examples	`rand(3.0) = 1.2937463`

gauss

Prototype	***double*** `gauss(`***double*** `m,` ***double*** `d)`
Description	`gauss(m,d)` returns normally distributed random number with mean value `m` and standard deviation `d`.
Examples	`gauss(0,2) = -.8678275`

round

Prototype	***double*** `round(`***double***`)` ***double*** `round(`***double*** `x,` ***double*** `y)`
Description	`round(x)` rounds `x` to the nearest integer. `round(x,y)` rounds `x` to the nearest multiple of y. Returns `x` if `y<=0`.

Examples	`round(1.5) = 2.0`
	`round(-1.5) = -1.0`
	`round(3.1415,0.1) = 3.1`

floor

Prototype	***double*** `floor(`***double***`)`
Description	Rounds down: finds the largest integer not greater than the argument, and returns it as a ***double***.
Examples	`floor(1.6) = 1.0` `floor(-1.6) = -2.0`

ceil

Prototype	***double*** `ceil(`***double***`)`
Description	Rounds up: finds the smallest integer not less than the argument, and returns it as a ***double***.
Examples	`ceil(1.6) = 2.0` `ceil(-1.6) = -1.0`

bool

Prototype	***bool*** `bool(`***bool***`)` ***bool*** `bool(`***int***`)` ***bool*** `bool(`***int64***`)` ***bool*** `bool(`***double***`)` ***bool*** `bool(`***complex***`)`
Description	Returns `false` if argument is equal to zero, returns `true` if argument is non-zero. `bool(x)` works exactly the same as type-casting operator `(bool)x`.

Examples	`bool(0) = false`
	`bool(1.5) = true`
	`bool(1.0+2.0j) = true`

bool *C-keyword*

Description	Declares boolean variable or array.

Examples	`bool b;`
	`bool var = false;`
	`bool array[10];`
	`bool array[] = { true, false, true };`

(bool) *type-casting operator*

Description	Declares boolean variable or array.

Examples	`bool b;`
	`bool var = false;`
	`bool array[10];`
	`bool array[] = { true, false, true };`

int

Prototype	***int*** `int(`***bool***`)` ***int*** `int(`***int***`)` ***int*** `int(`***int64***`)` ***int*** `int(`***double***`)` ***int*** `int(`***complex***`)`
Description	Returns argument value converted to `int` type. `int(bool x)` returns 0 if `x=false`, and returns 1 if `x=true`.`int(double x)` converts `double` to `int` by truncating (discarding the fractional part). `int(complex x)` converts `double` real part of a complex number to `int` by truncating (discarding the fractional part). `int(x)` works exactly the same as type-casting operator `(int)x`.

Examples	`int(true) = 1`
	`int(1.6) = 1`
	`int(-1.6) = -1`
	`int(1.1+2.2j) = 1`

int *C-keyword*

Description	Declares integer variable or array.

Examples	`int i;`
	`int var = 10;`
	`int array[10];`
	`int array[] = { 0, 1, 2, 3 };`

(int) *type-casting operator*

Description	Converts a number of arbitrary type to an integer. `(int)x` works exactly the same as function int`(x)`.
Examples	`i = (int)x;`

int64

Prototype	***int64*** `int64(`***bool***`)` ***int64*** `int64(`***int***`)` ***int64*** `int64(`***int64***`)` ***int64*** `int64(`***double***`)` ***int64*** `int64(`***complex***`)`
Description	Returns argument value converted to `int64` type. `int64(bool x)` returns 0i64 if `x=false`, and returns 1i64 if `x=true`.`int64(double x)` converts `double` to `int64` by truncating (discarding the fractional part). `int64(complex x)` converts `double` real part of a complex number to `int64` by truncating (discarding the fractional part). `int64(x)` works exactly the same as type-casting operator `(int64)x`.

Examples	`int64(true) = 1i64`
	`int64(1.6) = 1i64`
	`int64(-1.6) = -1i64`
	`int64(1.1+2.2j) = 1i64`

int64 *C-keyword*

Description	Declares 64-bit integer variable or array.

Examples	`int64(true) = 1i64`
	`int64(1.6) = 1i64`
	`int64(-1.6) = -1i64`
	`int64(1.1+2.2j) = 1i64`

(int64) *type-casting operator*

Description	Converts a number of arbitrary type to an 64-bit integer. `(int64)x` works exactly the same as function int64`(x)`.
Examples	`i = (int64)x;`

double

Prototype	***double*** double(***bool***) ***double*** double(***int***) ***double*** double(***int64***) ***double*** double(***double***) ***double*** double(***complex***)
Description	Returns argument value converted to `double` type. `double(bool x)` returns 0.0 if `x=false`, and returns 1.0 if `x=true`. `double(complex x)` returns real part of a complex number `x`. `double()` works exactly the same as type-casting operator `(double)`.

Examples	`double(true) = 1.0`
	`double(1) = 1.0`
	`double(1.1+2.2j) = 1.1`

double *C-keyword*

Description	Declares double variable or array

Examples	`double x;`
	`double var = 12.34;`
	`double array[10];`
	`double array[] = { 1.2, 3.4, 5.6 };`

(double) *type-casting operator*

Description	Converts a number of arbitrary type to a double. `(double)x` works exactly the same as function double`(x)`.
Examples	`x = (double)1/2;`

complex

Prototype	***complex*** complex(***bool***) ***complex*** complex(***int***) ***complex*** complex(***int64***) ***complex*** complex(***double***) ***complex*** complex(***complex***)
Description	Returns argument value converted to `complex` type. `complex(bool x)` returns 0.0 if `x=false`, and returns 1.0 if `x=true`. `complex(x)` works exactly the same as type-casting operator `(complex)x`.
Examples	`complex(true) = 1.0+0j` `complex(2) = 2.0+0j`

complex *C-keyword*

Description	Declares complex variable or array.

Examples	`complex c;`
	`complex var = 1.2+3.4j;`
	`complex array[10];`
	`complex array[] = { 1.0, 1.0j, -1.0, -1.0j };`

(complex) *type-casting operator*

Description	Converts a number of arbitrary type to a complex. `(complex)x` works exactly the same as function complex`(x)`.
Examples	`x = sqrt((complex)(-1));`

Script commands

In alphabetical order.

ac

Usage	`ac;`
	`ac` *`from`*`;`
	`ac` *`from`*`,` *`to`*`;`
	`ac` *`from`*`,` *`to`*`,` *`points`*`;`
	`ac` *`from`*`,` *`to`*`,` *`points`*`,` *`scale`*`;`

Description	Set AC analysis parameters and perform AC analysis. *`from`* : start frequency *`to`* : stop frequency *`points`* : number of points *`scale`* `= log` or `lin` : logarithmic or linear frequency scale. If called from the script, command will not return until AC analysis is completed. If called from console or HTTP link, returns immediately. Use `ready` command to check for analysis completion.

Examples	`ac;`
	`ac 1M;`
	`ac 1M, 100M;`
	`ac 1M, 100M, 500;`
	`ac 1M, 100M, 500, lin;`

clear

Usage	`clear;`
Description	Clear storage.

close

Usage	`close;`
Description	Close active document.

cmd

Usage	`cmd` *`command_line`*`;`
Description	Execute Windows command line *`command_line`*.
Examples	`cmd nl5.exe rc.nl5;` `cmd "C:\Arduino\arduino.exe" --upload "C:\Arduino\demo\demo.ino";`

cont

Usage	`cont;`
	`cont` *`screen`*`;`
	`cont` *`screen`*`,` *`step`*`;`

Description	Continue transient. *`screen`* : screen size *`step`* : calculation step If called from the script, command will not return until transient is completed. If called from console or HTTP link, returns immediately. Use `ready` command to check for transient completion.

Examples	`cont;`
	`cont 1m;`
	`cont 1m, 10n;`

cursors

Usage	`cursors` *`left`*`,` *`right`*`;`
	`cursors on;`
	`cursors off;`

Description	`cursors` *`left`*`,` *`right`* : set cursors (transient or AC) to specified positions and show cursors. *`left`* : position of the left cursor *`step`* : position of the right cursor `cursors on` : show cursors. `cursors off` : hide cursors.
Examples	`cursors 1.5, 2.5;` `cursors off;`

display

Usage	`display on;` `display off;`
Description	`display on` : show transient and AC windows. `display off` : hide transient and AC windows.

exit

Usage	`exit;`
Description	Close all documents and exit NL5. Cannot be called from console command line.

export *(transient)*

Usage	`export;`
	`export filename;`
	`export filename, from;`
	`export filename, from, to;`
	`export filename, from, to, step;`

Description	Export transient traces into csv file. *filename* : name of the file to export traces *from* : start of the data interval *to* : end of the data interval *step* : time step If *filename* is omitted, name of the file to export is the same as script file name, with "csv" extension. If file path is not specified, export in the script file directory. Extension "csv" can be omitted. Number of points cannot exceed **Max number of points** value defined in the **Preferences** dialog box, **Transient** page. If *step* is omitted, 101 points will be exported. Only traces currently shown on the graph will be exported.

Examples	`export;`
	`export rc_traces;`
	`export rc_traces, 0, 100;`
	`export rc_traces, 0, 100, 0.1;`

export *(AC)*

Usage	`export;`
	`export filename;`
	`export filename, from;`
	`export filename, from, to;`
	`export filename, from, to, points;`
	`export filename, from, to, points, scale;`

Description	Export AC traces into csv file. *filename* : name of the file to export traces. *from* : start frequency. *to* : end frequency. *points* : number of points. *scale* = `log` or `lin` : logarithmic or linear frequency scale. If *filename* is omitted, name of the file to export is the same as script file name, with "csv" extension. If file path is not specified, export in the script file directory. Extension "csv" can be omitted. Only traces currently shown on the graph will be exported.

Examples	`export;`
	`export ac_traces;`
	`export ac_traces, 1m, 1k;`
	`export ac_traces, 1m, 1k, 100;`
	`export ac_traces, 1m, 1k, 100, lin;`

import *(transient)*

Usage	`import filename;` `import filename, cf, cn, rf, rn, hr, tc, ts;`
Description	Import transient traces from text file or scope data file. *filename* : name of the file to import traces (with extension). *filename* should contain extension to specify type of data file. The following extensions/types are supported: "txt" and "csv" – text file, comma-separated. "wfm" – Tektronix waveform format. "isf" – Tektronix interval format. "bin" – Keysight Technologies (Agilent) binary format. "trc" – LeCroy binary format. More parameters can be used for import from text (comma-separated) files only. *cf* : first data column to import (1 is first column of the file). *cn* : number of data columns to import. If *cn* = -1, import all available columns after first. *rf* : first data raw to import (1 is first row of the file). *rn* : number of data rows to import. If *rn* = -1, import all available rows after first. *hr* : header row. If *hr* = 0, add header row: "*trace(s),trace1,trace2,…* " *tc* : time column. If *tc* = 0, add time column with time step *ts*. *ts* : time step of added time column floating point (set any value if not used).
Examples	`import scope_traces.isf;` `import rc_traces.csv, 2, -1, 2, -1, 1, 1, 0;`

logdata

Usage	`logdata` *`filename, expr1,...;`* `logdata` *`+, filename, expr1,...;`* `logdata;`
Description	`logdata` with parameters is the first data logging command. *`filename`* : name of the file to export traces *`+`* : flag to append the data into existing file *`exprN`* : expression to be logged If a file *`filename`* does not exist, creates a new log file and writes a header. If a file *`filename`* already exists, and a first parameter is *`+`*, a new data will be appended to existing data, otherwise old data will be overwritten. Extension "csv" in the file name can be omitted. If file path is not specified, creates log file in the script file directory. `logdata` without parameters evaluates expressions *`exprN`* specified in the first `logdata` command and writes results into the log file as comma-separated string.

Examples	`logdata rclog, r1, v(r1), v(c1).rms;`
	`logdata +, rcapp, r1, v(r1), v(c1).rms;`
	`logdata;`

open

Usage	`open` *`filename;`*
Description	Open schematic file *`filename`*. Extension "nl5" can be omitted. If file path is not specified, search in the script file directory.
Examples	`open "c:\Project files\nl5\rc.nl5";` `open rc;`

pause

Usage	`pause;`
Description	Pause transient. Command can be called from console command line and HTTP link only.

ready

Usage	`ready;`
Description	Check if transient or AC analysis is completed. Returns "0" if analysis is still running, returns "1" if completed. Command can be called from console command line and HTTP link only.

return

Usage	`return;`
Description	Stop executing the script
Examples	`return;`

rununtil

Usage	`rununtil;` `rununtil` *`expr`*`;`
Description	Set up "run until" transient mode. If parameter *`expr`* is omitted, turn off "run until" mode and clear "run until" expression. Otherwise turn on "run until" mode and use parameter *`expr`* as "run until" expression.
Examples	`rununtil;` `rununtil V(C1)<0;`

save

Usage	`save;` `save` *`filename`*`;`
Description	Save schematic into a file *`filename`*. Extension "nl5" can be omitted. If file path is not specified, save in the script file directory. If parameter *`filename`* is omitted, save into the same file.
Examples	`save;` `save rcnew;`

savedata

Usage	`savedata;` `savedata` *`filename`*`;`
Description	Save traces into "nlt" data file. Extension "nlt" can be omitted. If parameter *`filename`* is omitted, name of the file to save data is the same as script file name, with "nlt" extension. If file path is not specified, save in the script file directory. Only traces currently shown on the graph will be saved.
Examples	`savedata;` `savedata rctraces;`

saveic

Usage	`saveic;`
Description	Save Initial Conditions (IC).

scope.cmd

Usage	`scope.cmd` *`command`*`;`
Description	Send command *command* to the scope, returns scope response.
Examples	`scope.cmd :CHAN1:LABEL?;`

scope.get

Usage	`scope.get` *`number`*`;`
Description	Get a name of an instrument number *`number`*, *`number`* = 0...*number_of_instruments*-1

scope.getn

Usage	`scope.getn;`
Description	Get number of instruments.

scope.image

Usage	`scope.image;`
Description	Read scope screen image.

scope.log

Usage	`scope.log;`
Description	Read content of the Log tab of Scope window.

scope.off

Usage	`scope.off;`
Description	Close Scope tool window.

scope.on

Usage	`scope.on;`
Description	Open Scope tool window, refresh instruments list. All scope script commands can be performed with scope window closed, however it must be opened at least once in order to load VISA Library.

scope.read

Usage	`scope.read;`
Description	Read traces from the scope.

scope.refresh

Usage	`scope.refresh;`
Description	Refresh instruments list.

scope.run

Usage	`scope.run;`
Description	Run the scope in continuous mode.

scope.select

Usage	`scope.select` *`number`*`;`
Description	Select an instrument number *`number`*, *`number`* = 0...*number_of_instruments*-1

scope.single

Usage	`scope.single;`
Description	Run the scope in single mode.

scope.status

Usage	`scope.status;`
Description	Get text from the Scope window status bar.

scope.stop

Usage	`scope.stop;`
Description	Stop the scope.

scope.update

Usage	`scope.update;`
Description	Update scope configuration (read settings from the scope, update Scope window controls).

show

Usage	`show` *`window`*`;`
Description	Show or activate window specified by parameter *`window`*. The following *`window`* values can be used: `tran` : transient; `ac` : AC; `dc` : DC sweep; `xy` : XY diagram; `ed` : Eye diagram; `ah` : Amplitude histogram; `fft` : FFT; `th` : Transient histogram; `pow` : Power; `smith` : Smith chart; `ny` : Nyquist plot; `ach` : AC histogram.
Examples	`show tran;`

silent

Usage	`silent on;` `silent off;`
Description	`silent on` : do not show script execution log. `silent off` : show script execution log.

sleep

Usage	`sleep` *`time`*`;`
Description	Pause script execution for *`time`* ms.
Examples	`sleep 1000;`

stop

Usage	`stop;`
Description	Stop transient. This command can be used to free memory allocated for transient analysis. Transient cannot be continued after this command.

store

Usage	`store;` `store` *`expr`*`;`
Description	Move run into storage. The parameter *`expr`* is evaluated as an expression, and the result is used as a storage name. If parameter *`expr`* is omitted, a default storage name "RunN" is used.
Examples	`store;` `store R1*C1;`

storetext

Usage	`storetext;` `storetext` *`text`*`;`
Description	Move run into storage with parameter *`text`* as a storage name. If parameter *`text`* is omitted, a default storage name "RunN" is used.
Examples	`storetext;` `storetext This is a first run;`

traces

Usage	`traces` *`stateN,...`*`;`
Description	Hide or show traces on the graph. The parameter *`stateN`* specifies show/hide status of the trace number N (traces are listed in the same order as in the Transient/Data or AC/Data window). *`stateN`* = 0 – hide trace; otherwise – show trace.
Examples	`traces 0,1,1,0,0,1;`

tracename *(transient)*

Usage	*tracename;* *tracename from;* *tracename from, to;* *tracename from, to, step;*
Description	Request transient trace data as a comma-separated string. *from* : start of the data interval. *to* : end of the data interval. *step* : step. *tracename;* - returns 101 points of entire *tracename* interval. *tracename from;* – returns only one trace value at t=*from*. *tracename from, to;* – returns 101 points in specified interval. *tracename from, to, step;* – returns data points in specified interval with specified step. Trace *tracename* should be specified in the Transient Data, however it does not need to be displayed on the graph or in the table. Please note that length of the returned string may be limited, and the limit may be different for different applications. If you got a time-out on this command, please reduce the number of data points requested by one command. This command can be called from HTTP link only.
Examples	V(R1); V(R1) 1.23; V(R1) 0, 100; V(R1) 0, 10, 0.1;

tracename *(AC)*

Usage	*tracename*; *tracename from*; *tracename from, to*; *tracename from, to, points*; *tracename from, to, points, scale*;
Description	Request AC trace data as a comma-separated string. *from* : start frequency. *to* : end frequency. *points* : number of points. *scale* = log or lin : logarithmic or linear frequency scale. *tracename*; - returns all calculated data points of *tracename* trace. *tracename from*; – returns only one trace value at f=*from*. *tracename from, to*; – returns all calculated data points in the specified interval. *tracename from, to, points*; – returns specified number of points in the specified interval. *tracename from, to, points, scale*; – returns data with specified scale type. Trace *tracename* should be specified in the AC Data, however it does not need to be displayed on the graph or in the table. Please note that length of the returned string may be limited, and the limit may be different for different applications. If you got a time-out on this command, please reduce the number of data points requested by one command. This command can be called from HTTP link only.
Examples	V(R1); V(R1) 12.34; V(R1) 1, 100; V(R1) 1, 10, 100; V(R1) 1, 10, 100, lin;

tran

Usage	`tran;`
	`tran` *`start`*`;`
	`tran` *`start, screen`*`;`
	`tran` *`start, screen, step`*`;`

Description	Set transient parameters and start transient. *`start`* : start of transient display *`screen`* : screen size *`step`* : calculation step If called from the script, command will not return until transient is completed. If called from console or HTTP link, returns immediately. Use `ready` command to check for transient completion.

Examples	`tran;`
	`tran 0;`
	`tran 0, 10m;`
	`tran 0, 10m, 1u;`

Script examples

Set component parameters. Component parameters have been calculated in external application (for instance, Excel), or entered manually and saved into the text file in the ***name*****=*****value*** format:

```
R1 = 5.1;
C1 = 12e-9;
V3.period = 0.01;
```

Run the script to apply new parameters to components.

Sweep parameter. Component parameter is changing in specified range, transient analysis performed for each parameter, results placed into storage:

```
for( R1=1; R1<=10; R1+=1 )
{
  tran;
  store R1;
}
```

Sweep parameter from the list. Component parameter is assigned value from the list, AC analysis performed for each parameter, results placed into storage:

```
for( V1.period = 1m, 2m, 10m, 50, 100m )
{
  ac;
  store V1.period;
}
```

Sweep variable. Local variable is changing in some range, component parameters modified, transient analysis performed, results placed into storage:

```
double freq;
for( freq=1; freq<=10; freq*=1.1 )
{
  V2.period = 1 / freq;
  R2 = 1 / (freq * C5);
  tran;
  store freq;
}
```

Wait for condition. Transient is running until peak-to-peak value of the trace is less than specified threshold. When done, Initial Conditions are saved.

```
double threshold = 1e-6;
tran;
while( V(C1).pp > threshold ) cont;
saveic;
```

Perform analysis for specified file, save data, exit application. Schematic file is loaded into NL5, component parameters changed, transient analysis performed, traces exported into "csv" file, NL5 closed. This script can be executed from command line.

```
open lcr.nl5;
```

```
R1=100;
C1=1n5;
tran;
export data.csv;
exit;
```

Perform analysis for specified file, log data, exit application. Schematic file is loaded into NL5, component parameter swept, transient analysis performed, traces data logged into text file, NL5 closed. This script can be executed from command line.

```
open lcr.nl5;
logdata lcrdata.csv, r1, V(R1).mean, V(R1).rms;
for( R1=100; R1<=1000; R1+=100 )
{
  tran;
  logdata;
}
exit;
```